理论力学计算机模拟

彭芳麟　管　靖　胡　静　卢圣治　著



清 华 大 学 出 版 社

内 容 提 要

本书介绍数学软件 MATLAB 在理论力学教学中的应用。全书分为两章，第 1 章简明地介绍 MATLAB 的基础知识，第 2 章将理论力学教材中精选的 24 个典型的线性与非线性问题进行数值求解，给出了求解过程与程序，并将结果做成模拟实物运动的动画。本书用 MATLAB 对理论力学教材中的微分方程进行数值求解，有一定的独创性。书中大量的程序实例不仅实用，也有许多使用 MATLAB 的独特技巧，值得借鉴。全书将数值计算、学习 MATLAB 与学习理论力学融为一体，颇有新意，叙述从入门起步，循序渐进，读者只要具备大学理工科二年级的知识就可以看懂本书。本书既可作为大学选修课的教材，也可作为学习理论力学的参考书，还可作为学习 MATLAB 编程技巧的参考书。本书附送光盘一张，其上有书中的全部程序和一个多媒体课件，可供教师在课堂上作教学演示之用。此外，光盘上还有两段录像，分别介绍和演示 MATLAB 的基本用法及多媒体课件的使用。



前　言

计算机是信息时代最重要的标志之一。计算机及计算方法的高速发展使得计算机模拟成为物理研究中除实验、理论分析之外的第三种研究手段。这种变化势必会反映到大学物理课程的教学中来。又由于日新月异的计算软件的优秀性能，使得学习计算技术也成为一项轻松愉快的工作。原来在大学理论物理的教学中，对不能求解析解的常微分方程和偏微分方程是不要求学生掌握其解法的，今天借助先进的数学软件，我们已经可以很方便地教会学生如何去求解这些方程。学生在掌握这些解法之后，不仅提高了能力，也能更深刻地理解其中的物理规律。换言之，在教学中已经有可能把理论物理的学习与培养学生用计算机通过数值计算来研究物理问题的能力结合起来。

《理论力学计算机模拟》就是上述变化的产物，它是我们为北京师范大学物理系二年级本科生新开设的一门课程，该课程与理论力学课并行开设，教学安排是：先用 8 学时为学生介绍 MATLAB 的基本知识，然后安排学生每周 2 学时的上机，让学生完成一定数量的实验题目。期末再布置一些题目或由学生自己选题作为考察成绩之用。本书就是这门课程的教材，书中的程序，多数来自于学生的工作，再经过教师的加工整理而成。

开设这门课程的指导思想是：

1．理论力学教材现代化呼唤新的教学内容和新的教学手段

长期以来，理论力学教材的内容基本上是停留在牛顿理论和拉格朗日、哈密顿理论的框架内，数学上仅限于解析求解线性方程，这种状况已经不能适应当前的科技发展的形势。 20 世纪 60 年代取得突破并蓬勃发展起来的非线性科学，对当前和未来的科学发展起着重要的作用。新世纪的大学生急需更新观念，把目光引向科学的前沿——非线性科学。为此，作为教材现代化的一个重要内容，我们将非线性振动的基础知识作为介绍非线性科学的一个窗口，作为新的基础理论引入了新编的理论力学教材。

非线性科学的许多内容都是计算物理的重要成果，在非线性科学中遇到的大量的非线性微分方程都是用计算机数值求解的。在教材中介绍非线性振动，最好让学生直接学习用计算机的数值计算功能和可视化功能来揭示和探索非线性现象，这样做同时也解决了教材中原有的其他非线性问题。

2．教学手段和方法的现代化带来了新的教学目标

在当今的信息时代，计算机无孔不入，无处不在。计算机作为现代化的教学手段，已经广泛地应用于教学之中，但更多的是用作多媒体演示，是教师的教学辅助工具。如前所述，计算机模拟已经成了物理研究中的第三种手段，如

果把培养学生用计算机数值计算研究物理问题的能力作为新的教学目标，则计算机不仅仅是教师辅助教学的工具，也是学生学习的必备工具。与多媒体课件相比，让学生进行计算机模拟实验，使学生由观看变成参与，由被动接受变成主动研究，将更能激发学生的学习兴趣，调动学生的创造力，发挥学生的想象力，帮助学生更好地完成学习任务。但是如何去实现这一目标，则是见仁见智。我们在这门课程的做法是：将理论物理的学习与数值计算能力的培养结合起来，让学生用计算机来解决学习理论力学中遇到的计算问题，强调通过学用结合的实践来培养能力。

3．科学与工程计算语言 MATLAB 是数值计算的优秀工具

《理论力学计算机模拟》能够顺利开设的重要原因是因为采用了 MATLAB 作为计算工具。MATLAB 易学易用，对学生而言，它是一个真正的计算工具，而不是一门新的计算机课程。学生只要经过十多个小时的练习，就能用它完成所需要的计算，学生的精力是在研究物理问题上，而不是在编程计算上。所以在教学上是以学生探索式自学与上机实践为主，教师讲授为辅。在基本上没有增加学生负担的前提下，用很少的学时完成教学内容。学生学习的内容包括用 MATLAB 数值求解常微分方程并作时间历程图与相图；作快速傅里叶变换并画功率谱图；作符号运算如用拉普拉斯变换法解常微分方程组；作动画模拟与作用户界面等。书中的程序为 MATLAB 的编程应用提供了许多实际的技巧和方法。由此联想到，数学软件在物理中的应用日趋广泛，如何在物理课程的教学中反映出来，也是值得人们思考的问题。

本书的部分内容在 2001 年 7 月由理论力学教学研究会与北京师范大学物理系联合举办的全国高校理论力学教师讲习班上曾作过介绍，受到 30 余所高校教师的一致好评。在中国物理学会举办的全国多媒体与网络物理教学成果评比大会上本书的程序获得一等奖。在 2001 年第 8 、 10 、 11 期《大学物理》杂志上都有文章介绍这项工作。在中国物理学会教学委员会七届二次扩大会议上，这项工作的报告也引起了专家们的广泛兴趣和重视。我们希望通过本书的出版，能与更多的教师和学生进行交流，也能得到更多的专家和同行们的批评指导。由于这项工作还在探索之中，作者的学识水平有限。加之时间仓促，书中疏漏之处乃至错误难以完全避免，欢迎读者提出意见，我们的电子邮件地址是 ppffll@263.net 。

如果使用本书作为选修课教材，作者建议，教师只须讲解第 1 章的基本内容和第 2 章中几个简单的例子，如粒子散射、水星近日点的进动和落体偏东等，然后由学生自己完成 4~5 个例子，如阻尼斜抛运动、带电粒子在电磁场中的运动、小环在转动大环上的运动、傅科摆、滑动摆和倒摆的强迫振动等，就可以认为本课程的教学要求已经达到。其余的例子则作为选题，供教师考察学生能

力之用。学生可以根据兴趣从中选择一个例子加以研究，对程序进行修改甚至重新编程。当然学生也可以利用学到的知识去解决教科书中其它的习题。书中的程序仅供参考，它不是唯一的解法甚至不一定是最好的解法。读者完全可以独立地去解决书中的题目。我们的工作只是抛砖引玉，希望读者能编出更优秀的程序。

本书附有光盘，光盘中的 readme 是使用说明。光盘的 dyz 目录下是书中第 1 章中各个例题的程序，每个程序文件的命名与书中相同。光盘的 lxsy 目录下是书中的 24 个实验题目的程序，每个程序文件的命名也与书中相同。在 lxsyjm 目录下是作成了用户界面的 24 个实验题目的程序，这样即使不了解程序和编程过程，也能方便地使用这些程序作为多媒体演示的课件。光盘中还有两段教学录像，分别介绍 MATLAB 的基本用法及多媒体课件的用法。使用本书时，不妨先看看光盘上的录像，再实际演示一下多媒体课件中的各个程序，这样就会对本书内容有一个大致的印象。然后去阅读书上的内容，就很容易达到事半功倍之效。

北京师范大学物理系九九届学生在学习本门课程时所表现出来的巨大热情是本书成功的重要因素，其中吴海诚、江进武、刘志强、刘微、李光蕊、欧阳敏、胡进良、王平、闫静、龚雪飞、李昱、刘艳丽、尤文龙等同学表现十分突出，尤其是陈锡辉和贾砚宾两位同学直接参与了本书程序的整理与录入。还有九八届的沈福根、汤仙明、孙永新、吴国先、曹周健等同学曾参与本书的初期准备工作。

本书的分工是：彭芳麟编写第 1 章与第 2 章的程序并负责全书的组织、校对及将全书用 CCT 系统排版，管靖、胡静和卢圣治负责第 2 章中力学实验题目的编写。

作者十分感谢清华大学出版社对本书出版的支持，尤其感谢责任编辑宋成斌和韩燕丽为本书所付出的辛勤劳动。

著 者

2001 年 12 月




BEIJING NORMAL UNIVERSITY
1902

目　录

第 1 章　MATLAB 简介

本章介绍一些 MATLAB 的基本知识，内容侧重在物理计算中会用到的知识，如数据结构、有关计算的指令、编程以及图形与动画的制作。

1.1　预备知识

MATLAB 是矩阵工作室（ Matrix Laboratory ）的缩写，它是一个在欧美十分流行的通用性很强的优秀数学软件，占据了数学计算软件市场的主导地位。科技人员和工程人员即使不掌握 C 语言或 FORTRAN 语言，只要经过很短时间的学习，也能利用它轻松地去完成专业的科学计算和作图的任务。它不仅是科研的工具，也是高校里的本科生、硕士和博士们学习的好帮手。

MATLAB 的基本特点有：

①极强的数值计算功能和作图功能（可以作动画和用户界面），也有很强的符号计算功能。

②图形窗口式的操作界面，一看就懂，一用就会。使用常用的数学符号和表达式，贴近人们的思维习惯。默认使用复数与矩阵，计算速度快。

③用简单的指令就可以完成大量的计算与作图功能。编程语法简单，程序设计方便。绝大部分指令的程序是开放的，用户可以模仿和修改。

④有大量的不同领域的专用工具箱，用户还可以开发自己的专用工具箱。

本书选用 MATLAB 作为计算工具，就是因为用数学软件编程，简单明了，一目了然，初学者很快可以掌握编程的思路和方法。本书的全部程序都是用 MATLAB 编写的，从中可以看到， FORTRAN 要用数千行的语句编写的程序在 MATLAB 的程序中只是寥寥数行像公式一样简单明了的语句。学习和使用 MATLAB, 省时省力，可以把科技工作者从繁重的编程劳动解放出来，使他们有更多的时间和精力去研究物理规律本身。

本章的内容按教学顺序安排。对某些较简单的内容，本书只是略加介绍，有兴趣的读者可以参看本书参考文献中介绍的参考书。而对一些重要的又容易令初学者感到困惑的 MATLAB 的用法，如正确区分数据的类型，解非线性方程的指令 fsolve 中优化选项 (options) 的用法，解常微分方程中函数文件 (odefile) 的编写，指令的选用等，则尽量详细解释。其中有的是笔者自己应用 MATLAB 的心得体会。所选的例子大多数是物理问题，每题都带有参考程序 。笔者认为，

实际程序中的技巧，往往难以用文字解释。根据笔者的教学经验，读者在学完这些内容之后，已经能应用 MATLAB 来解决本书中的物理问题了。

在掌握 MATLAB 的基本操作以后，可以顺序阅读后面的内容，也可以从中选择一个自己感兴趣的课题去尝试编程解决它。在独立编程遇到困难以后再参考书中的程序，这比只是将书中程序在计算机上演示一遍所能得到的收获要大。编程能力的高低反映了掌握 MATLAB 的实际水平。只有经过独立编程的练习才能真正驾驭 MATLAB，所以读者一定要学会独立编辑一些程序。

本书内容是以 MATLAB 6.0 版本为准，与以前的版本相比，该版本增加了许多人机交互工作界面，使用起来更加方便快捷。但是 MATLAB 6.0 与 MATLAB 5.3 的指令变化不是太大，所以书中的程序在 MATLAB 5.3 的版本中也能运行。

1.1.1 操作界面

1. 操作桌面窗口与指令窗口

在 WINDOWS 操作系统的窗口中用鼠标双击 MATLAB 的图标即可进入如图 1.1 所示的 MATLAB 的操作桌面窗口 (Desktop)。如果没有图标，可利用 matlab \bin\win32 目录下的 matlab.exe 文件建立一个。

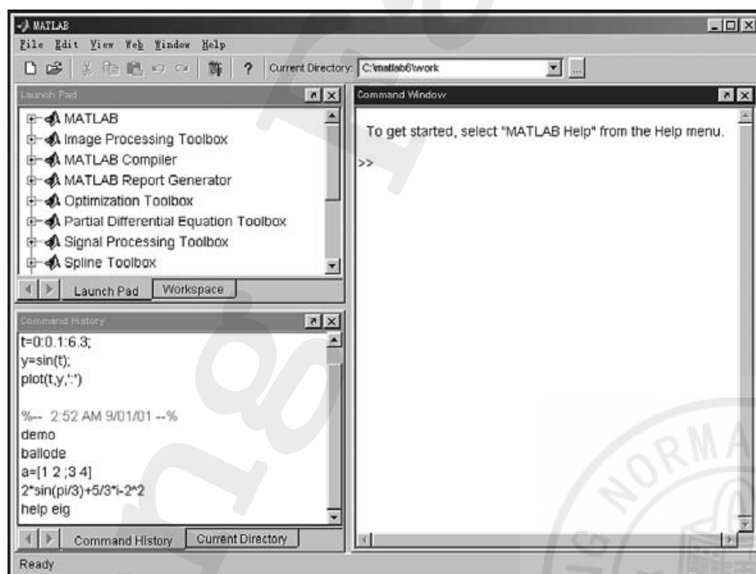


图 1.1 MATLAB 操作桌面窗口

这个窗口中包含有五个窗口，默认的设置有三个显示出来，另外两个是隐藏的。显示出来的是指令窗 (Command Window)、工作目录窗口 (Launch Pad) 和

指令记录窗口 (Command History)；隐藏的是当前工作的内存空间浏览器窗口 (Workspace) 和当前工作目录窗口 (Current Directory)。要使五个窗口都显现，可以依次逐层点击操作桌面窗口中的菜单 View\Desktop\Five Panel; 恢复默认设置则依次逐层点击 View\Desktop\Default; 由于大多数操作都是在指令窗进行，可以只显现指令窗，方法是依次逐层点击 View\Desktop\Command Window only。对于 MATLAB 6.0 以前的版本，操作界面就是指令窗。

要退出 MATLAB 的方法有三种：点击操作桌面窗口的右上角的 × 号，用菜单 File 下的 Exit MATLAB 或者直接在指令窗中键入 quit 后回车。

操作桌面窗口是 MATLAB 提供给用户的操作界面， MATLAB 所有的功能都是通过在操作桌面上各个窗口的操作来实现的。所以，操作桌面窗口是 MATLAB 的“大门”，初学者应该熟悉操作桌面内各个窗口的各种操作。

2. 指令窗的使用

在指令窗内可以进行数值计算、关系运算和逻辑运算或者调用 MATLAB 的各种函数指令和程序。

例如指令 demo 是演示 MATLAB 的基本功能，在指令窗中的指令行提示符“ >> ”后键入

```
>> demo
```

再按下 enter 键，就会显示 MATLAB Demos 窗口，再选择其中的各个条目并随时阅读窗口内的说明，就可以浏览 MATLAB 基本功能。这种浏览往往是初学者的入门捷径。注意指令行提示符“ >> ”是在 MATLAB 6.0 的版本中才有的，是自动显现在窗口中的，以前的版本则没有，绝不可以自己输入一个指令行提示符“ >> ”。

又如指令 ballode 是自由下落的小球和地面发生非弹性碰撞时的运动轨迹，键入

```
>> ballode
```

再按下 enter 键，就会显示一个图形窗口，窗口内图形画出了小球的运动轨迹。

要计算表达式 $2\sin\pi/3+5\mathrm{i}-2^2\div 3$ 的值，可键入

```
>> 2 * sin(pi/3) + 5*i - 2^2/3
```

按下 enter 键以后得到

```
ans =
      0.3987 + 5.0000i
```

这里 ans 是 answer 的略写，是 MATLAB 默认的输出变量名。

要输入矩阵 $\boldsymbol{A}=\begin{bmatrix}5 & 8\\ 2 & 6\end{bmatrix}$ 可键入

```
>> A = [ 5, 8; 2, 6 ]
```

得到

```
A =
    5  8
    2  6
```

3. 指令窗操作注意事项

MATLAB 使用的是双精度的计算，能自动识别复数，指令窗默认的显示格式是 5 位数字的定点表示，可以改变成其它表示格式。比如说，如果要改成 15 位数字的定点表示，可用鼠标双击菜单 File\Preferences，然后在对话窗口中选择 Command Window ，再选 Numeric format 条目中的 long 即可。这时在指令窗输入

```
>> pi
```

按下 enter 键以后得到的是

```
ans =
      3.14159265358979
```

如果在对话窗口中选择 General\Font & Colors，就可以改变字体种类、大小和颜色，等等。

对初学者，在指令窗操作时还应特别注意以下几点：

①所有输入的指令、公式或数值必须按下回车键以后才能执行。

②所有的指令、变量名称都要区分字母的大小写。

③应该指定输出变量名称，否则 MATLAB 将以 ans 作为默认的输出变量名。

④如果不需要在屏幕上显示结果，可以在命令后面加上分号，如 `A = 5 + 8`。

⑤如果要执行多个指令，可以将它们编辑为一个程序（方法见后面介绍）。在程序中，如果命令太长，在一行写不下，可以用三个圆点 ... 分行，表示命令延续到下一行。如

```
s = 1 - 1/2 + 1/3 -1/4 + 1/5 - 1/6 + 1/7 ...
    - 1/8 + 1/9 - 1/10 + 1/11 - 1/12
```

⑥等号 =、加号 + 和减号 − 前后的空格数不影响算符的作用。

1.1.2 在线帮助

MATLAB 有数千条指令，初学者不可能也没有必要等到学完并记住这些指令以后才开始使用 MATLAB。一般只须对 MATLAB 略有了解便可以使用它，然后边用边学，由浅入深，直到精通。这时最容易遇到的问题是，不了解 MATLAB 有哪些指令，也找不到自己想要的指令，当然更不会使用这些指令。另外，随着

MATLAB 的推广应用， MATLAB 的版本更新也在不断加快，功能不断增加，但各个版本的指令可能会有不同，随时掌握新版本的指令，也是用户要解决的问题。解决这些问题都可以借助 MATLAB 的在线帮助系统。 MATLAB 的在线帮助系统很全面，能使用户很方便地查阅到自己想要的信息，但不同版本的帮助系统用法有所不同。学会使用在线帮助系统，常常是学习一种新的软件的速成方法。

由于本章只介绍了一部分指令的用法，为使读者了解更多的指令，在附录 A 分类列出了 MATLAB 的一些主要的但不是全部的指令。排列的顺序是，在第一个条目“主要函数指令分类”中，列出了所有指令的分类目录，在每类下面都包含若干指令，其后的条目再将这些指令具体列出。浏览一下，对 MATLAB 会有一个粗略的了解，随时查阅，也很方便。

在指令窗使用帮助系统的最简单方法是键入

>> help + 指令名称

它能把一个指令的语法和功能的最基本信息直接显示在指令窗中，是最快捷的取得帮助的途径。本书中对许多指令只介绍名称而不介绍用法，所以使用它们之前可用这个指令查一下它们的用法。比如想查询指令 flipud 的用法，操作是

```
>> help flipud
```

得到

```
FLIPUD Flip matrix in up/down direction.
   FLIPUD(X) returns X with columns preserved and rows flipped
   in the up/down direction.  For example,
   X = 1 4      becomes  3 6
       2 5               2 5
       3 6               1 4
   See also FLIPLR, ROT90, FLIPDIM.
```

帮助系统告诉我们，这个指令的用法是将矩阵的各列上下翻转，并且举了一个例子来说明它的用法。此外，与这个指令有关的指令还有 fliplr, rot90, flipdim 。

注意， MATLAB 的在线帮助总是用大写字母来表示指令和变量名的（如上面的例子中指令是用 FLIPUD 表示），使其能与文字内容区分开来。然而使用指令名时必须使用小写（如在用 help 查找指令用法时用的是 flipud ），因为 MATLAB 是区分大小写的，而实际使用的是小写。

除了上述帮助系统的用法之外，依次点击菜单 Help\MATLAB help 将会打开一个 Help 窗口，你可以从中查找自己所需要的内容，此外 MATLAB 带有一份较完整的 PDF 格式的说明书，在 matlab\help\pdf-doc 文件夹中，用鼠标双击相应的文件名即可打开。

在指令窗中还有一些常用指令：

who	列出内存中变量名	type	显示指定文件的内容
whos	列出内存中变量名及其性质	which	列出文件所在的目录
clear	清除内存	dbtype	显示文件中带行号的内容
clc	清除工作窗中的显示内容	disp	显示文字或变量内容
clf	清除图形窗中的显示内容	edit	编辑指定的文件
what	查看指定目录下的文件名	cd	改变当前工作子目录
exist	查找变量或文件	dir	列出指定目录下的文件

此外，在指令窗中键入 !, 就能打开一个 DOS 窗口，在其中可以进行 DOS 环境下的各种操作，如直接调用外部程序等。

1.2 矩阵与表达式

本节介绍数据、变量名、算符和表达式的写法。

1.2.1 数据、变量名、算符与表达式

1. 数据格式

MATLAB 使用常规的十进制表示法，小数位数不限，可以用加号和减号表示正负数。10 的幂次表示为 e (或 E) 加上正负数字表示。MATLAB 默认的虚数单位是 i 和 j, 数字后面可以直接加上 i 或 j 表示虚数，中间不要有空格或乘号；但表达式中或与矩阵连用时要在 i 或 j 之前加上 * 号。以下都是合法的数值表示法：

$$\begin{array}{rrr} 3 & -99 & 0.00001 \\ 9.639728 & 1.062010\text{e}-20 & 6.02252\text{e}23 \\ 5+3\text{i} & -3.14159\text{i} & 3\text{e}5\text{i} \end{array}$$

2. 算符

MATLAB 使用人们所熟悉的算符，表 1.1 是算术运算算符，表 1.2 是关系运算算符，表 1.3 是逻辑运算算符。

表 1.1 算术运算算符

加	减	乘	除	幂	括 号
+	−	*	/	∧	( )

表 1.2　关系运算算符

小　于	小于等于	等　于	大　于	大于等于	不等于
<	<=	==	>	>=	~=

表 1.3　逻辑运算算符

与	或	非
&	|	~

3. 表达式

用算符把数字、变量和函数组合在一起，就建立了一个表达式。MATLAB 的数学表达式可以对整个矩阵进行运算。表达式中运算按常规的优先级自左至右执行，优先级的顺序是指数运算最优先，其次是乘除，最后是加减，可以用括号改变运算顺序。

4. 变量

可以用字母 (即变量名) 来代表具体的数据 (如矩阵)，这称之为对一个变量赋值。使用变量名不必说明其是否是复数，是多少维，精度是多少， MATLAB 总是将它们默认为双精度的复数。

变量名用字母打头，后面可以跟字母、数字、下划线，数目不限。但只有变量名的前面 31 个符号有效。要查看一个变量的内容，只需键入其名称。

在指令窗的所有变量都会保存在工作内存空间，只要不关闭指令窗就可以调用它们。用指令 whos 可以查看它们的名称。清除某个变量用指令 clear + 变量名，清除所有变量用指令 clear all 。

在 MATLAB 6.0 的操作桌面窗口中有一个工作内存空间窗口 (Workspace) ，变量的名称、维数、所占空间大小及类型都列在其中。点击变量的名称，就会再打开一个矩阵编辑器窗口 (Array Editor) ，变量的元素以表格形式排列其中，可以非常方便地查看和修改，也可以用它来输入大型矩阵。 MATLAB 6.0 以前版本没有操作桌面窗口，要用菜单 File\Show Workspace 来打开工作空间浏览器窗口 (Workspace Browser) 。

在 MATLAB 中某些变量名被默认为代表某些常用的常量：

pi　3.1415926　eps　2.2204e−016
i，j　虚数单位　realmin　最小浮点数 2.2251e−308
Inf　无穷大　realmax　最大浮点数 1.7977e+308
NaN　非数

无穷大 Inf 产生于不为零的数被零除，或被值溢出的数学表达式所除。比如，超过最大实数的数作除数。非数 NaN 产生于求表达式 (0/0) 或 (Inf−Inf) 的值，它们的数值没有定义。这些函数名并不是 MATLAB 保留的，它们可以重新定义为新的变量，例如可定义

```
>> eps = 1.0e-6
```

然后将其用于以后的运算。要恢复原来的内部函数值，可以键入

```
>> clear eps
```

5. *函数*

MATLAB 中有全部初等函数的指令，如绝对值、平方根、指数、三角函数和双曲函数 (见附录 A5.1)。取负数的平方根和对数会自动地得出相应的复数。MATLAB 也提供了许多更高级的函数，如贝塞尔函数和伽马函数 (见附录 A5.2)。大多数这些函数都能接受复数。要看基本函数列表，只需键入

```
>> help elfun
```

要列出特殊函数可以键入

```
>> help specfun
```

某些函数，比如方根、正弦函数，是内建的，是 MATLAB 的内核的一部分，所以非常有效，但计算的细节则不容易得到。另一些函数，如双曲正弦函数、伽马函数，是用 M 文件 (即扩展名为 .m 的文件) 来建立的。你可以查看文件的核心内容甚至修改它。

用这些基本函数可以建造自己专用的单变量或多变量函数，方法有三种：用指令 inline，用符号变量和用 M 文件建造函数。这里先介绍用指令 inline 建造函数的方法 (用符号变量建造函数的方法见 1.2.3 节，用 M 文件建造函数的方法见 1.3.2 节)。

要建立一个带参数 θ 的 x 的函数

$$ff = \cos^2 x^2 + \theta$$

并想得到当 $x = 3,\ \theta = 2.1$ 时函数的值。使用指令 inline 的操作如下：

```
>> FF = inline( 'cos(x^2)^2 + theta', 'x', 'theta' )
FF =
     Inline function:
     FF(x,theta) = cos(x^2)^2+theta
>> FF( 3, 2.1 )
ans =
      2.9302
```

如果变量 **x** 是矢量，则要将函数表达式中的算术运算算符改成对数组的运算算符（见 1.2.2 节），也可以用指令 vectorize 来完成这个任务，指令 vectorize 的功能是将公式矢量化，即在算符“∧”、“∗”、“/”的前面增加一个圆点“·”。两种方法举例如下。第一种操作是用指令 vectorize 将已有的函数 FF 矢量化：

```
>> vectorize(FF)
ans =
      Inline function:
      ans(x,theta) = cos(x.^2).^2+theta
```

第二种操作是在输入时使用对数组运算的算符：

```
>> FFF = inline( 'cos(x.^2).^2 + theta', 'x', 'theta' )
FFF =
      Inline function:
      FFF(x,theta) = cos(x.^2).^2+theta
```

现在可以求矢量形式的变量 **x** 对应的函数值。

```
>> x = 3 : 0.1 : 4;
>> FFF( x, 2.1 )
ans =
      2.9302   3.0661   2.5702   2.1111   2.3862   3.0032   2.9529
      2.2870   2.1889   2.8719   3.0171
```

由于在 MATLAB 中主要使用矩阵，建议表达式或函数都采用矢量化形式。

对已经建立的函数可用函数名查询，如键入 FFF 就可以查看刚才建立的函数的形式和它的变量。

要建立一个多变量函数，必须将各个变量表示成一个矩阵的分量。同样，多变量函数也可以带参数。下面建立一个带有参数 c 的三个变量 x, y, z 的函数。

$$gg = -x^2 + y^2 - z^2 + 6 + c$$

首先将 x, y, z 看成是 **v** 的三个分量。当 **x, y, z** 是矢量时，可以用 1.2.2 节的冒号算符表示成

```
v(:,1) = x,  v(:,2) = y,  v(:,3) = z
```

则 **v** 是有三个列矢量的矩阵。现在用 **v** 来建立函数。

```
>> gg = inline( '-v(:,1).^2 +v(:,2).^2 -v(:,3).^2 +6 +c', 'v', 'c' )
gg =
     Inline function:
     ff(v,c) = -v(:,1).^2 + v(:,2).^2 - v(:,3).^2 + 6 + c
```

再求当 $\boldsymbol{x} = [\,1, 3, 6\,]$，$\boldsymbol{y} = [\,2, 2, 7\,]$，$\boldsymbol{z} = [\,3, 1, 8\,]$ 时的函数值，

```
>> gg( [ 1 2 3; 3 2 1; 6 7 8 ], 3 )
ans =
     3
     3
   -42
```

这个结果仍是一个列矢量。

1.2.2 矩阵

MATLAB 的优势在于能快捷和轻松地处理整个矩阵，而其它的编程语言一次只能处理一个数据。所以要充分发挥 MATLAB 的优点，就应该尽量使用矩阵运算。

在 MATLAB 中把数据分为标量 (scalar) 、矢量 (vector) 、矩阵 (matrix) 和列阵 (array) 。列阵是指多维数组，所以列阵包括的范围最广。在列阵中，一维数组是矢量，二维数组就是矩阵，此外，在列阵中还有三维或四维数组等，甚至可以有字符串构成的列阵。对于不同的数据对象， MATLAB 的指令往往不同，即使同一个指令对不同的数据对象也可能有不同调用格式。

1. 定义矩阵

定义矩阵可以直接输入矩阵的各个元素或用 MATLAB 的指令函数生成。

在输入矩阵的各个元素时，要用空格或逗号分割同一行的各个元素，用分号结束一行元素，用方括号括起整个矩阵。定义矩阵不需要说明维数，可以直接使用复数。最简单的是列矢量或行矢量，如生成有 16 个元素的行矢量 $\boldsymbol{A}$ 的做法是在指令窗输入

```
>> A = [ 16  3  2  13  5  10  11  8  9  6  7  12  4  15  14  1 ]
A =
    16   3   2   13   5   10   11   8   9   6   7   12   4   15   14   1
```

要生成 4×4 的矩阵 $\boldsymbol{A}$ 则可以利用分号：

```
>> A = [ 16  3  2  13; 5  10  11  8; 9  6  7  12; 4  15  14  1 ]
A =
    16    3    2   13
     5   10   11    8
     9    6    7   12
     4   15   14    1
```

如果在每一个元素后面都加上分号，就会生成一个列矢量。

生成矩阵的指令有：

zeros	零矩阵	ones	全部元素为 1 的矩阵
eye	单位矩阵	rand	均匀分布的随机数矩阵
magic	幻方阵	randn	n 维正态分布的随机数矩阵
cell	空矩阵	diag	对角矩阵或提取对角元
linspace	等间距的矢量	logspace	对数等分的行矢量

试举数例如下：

```
>> Z = zeros ( 2, 4 )
Z =
    0   0   0   0
    0   0   0   0
>> F = 5*ones( 3, 3 )
F =
    5   5   5
    5   5   5
    5   5   5
>> R = randn( 4, 4 )
R =
    -0.4326   -1.1465    0.3273   -0.5883
    -1.6656    1.1909    0.1746    2.1832
     0.1253    1.1892   -0.1867   -0.1364
     0.2877   -0.0376    0.7258    0.1139
>> a=[ 1, 3, 5 ];
>> diag(a)
ans =
      1    0    0
      0    3    0
      0    0    5
>> diag( a, -1 )
ans =
      0    0    0    0
      1    0    0    0
      0    3    0    0
      0    0    5    0
>> diag( a, 1 )
ans =
```

```
        0    1    0    0
        0    0    3    0
        0    0    0    5
        0    0    0    0
>> cell(4)
ans =
        [ ]       [ ]       [ ]       [ ]
        [ ]       [ ]       [ ]       [ ]
        [ ]       [ ]       [ ]       [ ]
        [ ]       [ ]       [ ]       [ ]
```

2. 标识矩阵元素

标识矩阵元素是指标识某个、某行或某列元素。

(1) 行标与列标

在 MATLAB 编程中，矩阵元素 $\boldsymbol{A}_{ij}$ 表示为 A(i, j) ，矢量元素 $\boldsymbol{B}_k$ 表示为 B(k) 。矩阵元素也可以用一个指标来表示，这时是把整个矩阵看成一个列矢量，元素计数的顺序是第一列、第二列直到最后一列。如对于前面 4×4 的矩阵 $\boldsymbol{A}$, 矩阵元素 A(4,2) 可以表示为 A(8) 。注意，这两种表示法如果使用不当，会造成 MATLAB 的指令操作错误。

在查找矩阵元素 A(i, j) 时，如果 i, j 超出了矩阵行数和列数的范围，则显示出错信息。但是在存储矩阵元素 A(i, j) 时，如果 i ， j 超出了矩阵行数和列数的范围，则矩阵会自动扩充并以零值填补没有输入的元素。如在前面的矩阵 $\boldsymbol{A}$ 中增加一个元素 A(4, 5) ，得到的是

```
>> A(4, 5) = 17
A =
     16    3    2   13    0
      5   10   11    8    0
      9    6    7   12    0
      4   15   14    1   17
```

(2) 冒号算符

冒号算符 : 可用来生成矢量、矩阵或表示矩阵元素，它有几种不同的用法。从 1 到 10 步长（间距）为 1 的行矢量的表达式是

```
>> M = 1 : 10
M =
     1    2    3    4    5    6    7    8    9    10
```

如果步长不是 1 ，要指明步长，例如从 0 到 π 步长为 $\pi/4$ 的行矢量是

```
>> M = 0 : pi/4 : pi
M =
     0   0.7854   1.5708   2.3562   3.1416
```

带有冒号的指标表达式是用来表示矩阵的一部分。如 A(1:k, j) 表示矩阵 $\boldsymbol{A}$ 的第 j 列的前面 k 个元素。冒号 : 还可以代表一行的全部元素或一列的全部元素，如 A(:, j) 和 A(i, :) 分别是矩阵 $\boldsymbol{A}$ 的第 j 列和第 i 行。用 end 可以表示最后一个元素，如 A(end, j) 和 A(i, end) 分别是矩阵 $\boldsymbol{A}$ 的第 j 列和第 i 行的最后一个元素。

3. 修改矩阵

矩阵的修改包括矩阵的合并、删去矩阵中指定的行或列、对矩阵中的特殊元素加以过滤。

(1) 合并

合并是用一些小矩阵来建造大矩阵。其实前面建造矩阵就是将单个元素合并。方括号 [ ] 就是合并算符。例如用前面建立的 4×4 的矩阵 $\boldsymbol{A}$ 可以造成包含有四个子矩阵的 8×8 的矩阵

```
>> B = [ A  A+32; A+48  A+16 ]
B =
    16     3     2    13    48    35    34    45
     5    10    11     8    37    42    43    40
     9     6     7    12    41    38    39    44
     4    15    14     1    36    47    46    33
    64    51    50    61    32    19    18    29
    53    58    59    56    21    26    27    24
    57    54    55    60    25    22    23    28
    52    63    62    49    20    31    30    17
```

(2) 删去行或列

在方括号中如果没有元素就表示空矩阵，所以可以用方括号从矩阵中删去一行或一列。要删去矩阵 $\boldsymbol{A}$ 的第二列，做法是

```
>> A( : , 2 ) = [ ]
A =
    16     2    13
     5    11     8
     9     7    12
     4    14     1
```

如果想从矩阵中删去一个元素，则其不成为一个矩阵，所以如下的表达式是错误的

```
>> A( 1 , 2 ) = [ ]
```

用一个指标的表达式删去一个元素或一个元素序列，剩余元素将构成一个列矢量。所以

```
>> A( 2 : 2 : 10 ) = [ ]
```

得到的是

```
A =
    16     9     2     7    13    12     1
```

这里把矩阵看成列矢量，矢量的编号是从第一列，第二列，直到最后一列。所以上述命令表示从矩阵表示的列矢量中删去第 2， 4， 6， 8， 10 号元素。

(3) 逻辑下标

有时在矩阵中存在不合乎要求的数据，希望把它们过滤掉，但并不知道它们的确切位置和数量。这时可以利用具有逻辑运算功能的指令来完成这个任务。例如，已知数组

```
>> x = [2.1  1.7  1.6  1.5  NaN  1.9  1.8  1.5  5.1  1.8  1.4  2.2]
x =
 Columns 1 through 7
    2.1000  1.7000  1.6000  1.5000  NaN  1.9000  1.8000
 Columns 8 through 12
    1.5000  5.1000  1.8000  1.4000  2.2000
```

其中的 NaN 表示非数，是一个丢失的数据，为了去掉它，可以使用 finite(X) 。它对所有有限数，其值为真，对 NaN 和 Inf 为假。

```
>> x = x( finite(x) )
x =
 Columns 1 through 7
    2.1000  1.7000  1.6000  1.5000  1.9000  1.8000  1.5000
 Columns 8 through 11
    5.1000  1.8000  1.4000  2.2000
```

此时，数组中还有一个数 5.1 与其它的数偏离过大，它是一个无效数。可以用以下方法来去掉它。假定要去掉与数组的平均值的偏差大于标准差两倍的数，使用的指令是

```
>> x = x( abs( x - mean(x) ) <= 2 * std(x) )
x =
 Columns 1 through 7
```

```
   2.1000  1.7000  1.6000  1.5000  1.9000  1.8000  1.5000
Columns 8 through 10
   1.8000  1.4000  2.2000
```

4. 矩阵操作的指令和算符

(1) 三类指令

MATLAB 的算符和指令的形式都与作用对象有关，以指令为例，可分三类。下面以对矩阵 $\boldsymbol{A}$ 的作用为例加以说明。

对标量作用的指令叫标量函数，作用对象是矩阵 $\boldsymbol{A}$ 中的每一个元素，如 sin(A)，是对矩阵 $\boldsymbol{A}$ 的每一个元素求正弦值。

对矢量作用的指令叫矢量函数，作用对象是矩阵 $\boldsymbol{A}$ 中的每一列元素，如 max(A)，是对矩阵 $\boldsymbol{A}$ 中每列元素求极大值。

对矩阵作用的指令叫矩阵函数，作用对象是矩阵 $\boldsymbol{A}$ 中的全体元素，如 inv(A)，是对整个矩阵求逆。

从下面的例子可以看出这些指令之间的差别。例如求矩阵 B 的各个元素的正弦值的操作是

```
>> B = [ pi/6, pi/4; pi/3, pi/2 ];
>> sin(B)
ans =
      0.5000    0.7071
      0.8660    1.0000
```

又如求矩阵 $\boldsymbol{G}$ 的本征矢量 $\boldsymbol{X}$ 和对应的本征值 $\boldsymbol{V}$ 及各列的平均值的操作是

```
>> G = [ 1, 8, 4; 6, 8, 8; 3, 5, 8 ];
>> [ X, V ] = eig(G)
X =
    0.8920   -0.4681   -0.5321
   -0.4505   -0.7173   -0.4583
   -0.0378   -0.5161    0.7119
V =
   -3.2096         0         0
         0   17.6707         0
         0         0    2.5390
>> mean(G)
ans =
      3.3333    7.0000    6.6667
```



所以同一个命令作用在不同的对象上可以有不同的结果。如求和指令 sum 是对矩阵的每一列元素求和，对一个 $n \times m$ 的矩阵作用的结果是产生一个有 m 个元素的行矢量，而它对一个有 m 个元素的行矢量或列矢量作用的结果是得到一个数。又如矩阵求逆指令 inv 只能对二维的矩阵作用，如将它作用在三维的列阵上，则会给出错误的信息。

(2) 两种算符

算符可分为两种，一种是按矩阵运算法则定义的运算算符，叫矩阵运算算符；另一种是按矩阵的对应元素进行运算的算符，叫数组运算算符。两种运算的算符列在表 1.4 中。

表 1.4 矩阵运算算符与数组运算算符

	共轭转置	加	减	乘	右 除	左 除	幂
矩阵运算算符	A′	+	−	*	/	\	∧
数组运算算符	A.′	+	−	.*	./	.\	.∧

例如对矩阵 A 和 B 分别进行矩阵运算乘法和数组运算乘法的结果是

```
>> A = [ 15, 7, 5; 12, 3, 14; 7, 10, 11 ];
>> B = [  6, 7, 3; 11, 14, 13; 4, 10, 9 ];
>> C = A*B
C =
    187   253   181
    161   266   201
    196   299   250
>> D = A.*B
D =
     90    49    15
    132    42   182
     28   100    99
```

可以看到，其中的 C 是矩阵运算的结果。按照线性代数的定义， C 的第一个元素是 A 的第一行元素乘以 B 的第一列元素所得结果之和； D 是数组运算的结果，是 MATLAB 中定义的运算规则， D 的第一个元素是 A 的第一个元素乘以 B 的第一个元素所得的结果，两者结果大相径庭。

在 MATLAB 中定义的矩阵运算左除“ \ ”和右除“ / ”的含义是，如果 A * B = C ，则 B = A \ C, A = C / B 。注意，矩阵的维数要满足矩阵运算的条件。例如利用刚才的矩阵 A 、 B 和 C 可得

```
>> A \ C
ans =
      6.0000    7.0000    3.0000
     11.0000   14.0000   13.0000
      4.0000   10.0000    9.0000
>> C / B
ans =
      15.0000    7.0000    5.0000
      12.0000    3.0000   14.0000
       7.0000   10.0000   11.0000
```

所得结果分别是 B 和 A 。

常用的矩阵运算指令有：

sum	求矩阵各列元素之和	det	求矩阵行列式的值
prod	求矩阵各列元素之积	eig	求矩阵本征值与本征矢量
max	求矩阵各列的最大元素	fliplr	使矩阵左右翻转
min	求矩阵各列的最小元素	flipud	使矩阵上下翻转
mean	求矩阵各列的平均值	rot90	将矩阵逆时针转 90°
std	求矩阵各列的标准差	median	求矩阵各列的中位元素
abs	求复数的模	cumsum	求矩阵各列元素累计和
angle	求复数的辐角	cumprod	求矩阵各列元素累计积
fft	快速傅里叶变换	sort	按递增排列各列元素
ifft	快速傅里叶逆变换	rank	求矩阵的秩
trace	求矩阵的迹	cross	计算矢量叉积
kron	计算张量积		

上面只列举了一部分指令，附录 A 中有更多的指令介绍。 MATLAB 对绝大多数矩阵运算都有专门的指令，读者应该养成习惯，在进行矩阵运算时，先查一下有关的指令用法。

MATLAB 还可以把整个矩阵当作一个变量进行指数、对数等运算，这类指令有：

expm	计算矩阵指数	logm	计算矩阵对数
aqrtm	计算矩阵平方根	funm	以矩阵为变量的矩阵函数

从下面的例子可以看出，如果 $\mathrm{B} = \mathrm{logm(A)}$，则有 $\mathrm{A} = \mathrm{expm(B)}$ 。而 $\mathrm{b} = \log(\mathrm{A})$ 是对 A 的每一个元素求对数，所以 $\mathrm{b} \neq \mathrm{B}$ 。但是如果对 b 的每一个元素再求相应的指数运算 exp(b)，答案当然也是 A 。

```
>> A = [ 5, 7, 8; 12, 9, 11; 3, 2, 1 ];
```

```
>> B = logm(A)
B =
    1.6932 + 2.0173i   0.6564 - 1.0214i   0.5818 - 1.1292i
    1.3027 - 1.7913i   1.7134 + 1.5142i   1.8869 - 1.7991i
    0.0629 - 0.3882i   0.4560 - 0.3527i   0.6538 + 2.7517i
>> expm(B)
ans =
       5.0000 - 0.0000i   7.0000 - 0.0000i   8.0000 + 0.0000i
      12.0000 - 0.0000i   9.0000 - 0.0000i  11.0000 + 0.0000i
       3.0000 + 0.0000i   2.0000 + 0.0000i   1.0000 + 0.0000i
>> b = log(A)
b =
    1.6094    1.9459    2.0794
    2.4849    2.1972    2.3979
    1.0986    0.6931         0
>> exp(b)
ans =
      5.0000    7.0000    8.0000
     12.0000    9.0000   11.0000
      3.0000    2.0000    1.0000
```

1.2.3 符号变量

如前面所看到的那样，数值计算中用的变量都是已经赋值的。没有赋值的变量叫符号变量。在 MATLAB 中对符号变量的定义和运算都有专门的规定。因为最初的 MATLAB 是一个数值计算软件，后来利用数学软件 MAPLE 的符号计算功能开发了符号计算工具箱（ Symbolic Math Toolbox ）和扩展的符号计算工具箱（ Extended Symbolic Math Toolbox ），以便在 MATLAB 的环境下实现符号计算。1.6.1 节符号运算工具箱专门介绍这些内容。这里提及符号变量只是用来和数值运算作对比。

建立一个符号变量的命令是

```
>>x = sym('x')
```

而建立多个变量的命令是

```
>> syms  x y z a
```

它等效于

```
>> x = sym('x')
```

```
>> y = sym('y')
>> z = sym('z')
>> a = sym('a')
```

可以用上面定义的符号变量来建立符号表达式并进行运算，这也就是前面说的建立函数的第二种方法。如下面的操作是建立一个函数

$$g = \sin(x)/x$$

并用指令 limit 求函数在 $x = 0$ 的极限，以及用指令 sub 求 $x = 0.5$ 的函数值。

```
>> g = sin(x) / x
g =
    sin(x)/x
>> limit( g, 0 )
ans =
      1
>> subs( g, x, 0.5 )
ans =
      0.9589
```

如果不定义 x 为符号变量而直接输入 $\sin(x)/x$, 就会显示出错的信息。显然符号运算与数值运算是完全不同的概念。符号变量的定义可以灵活应用，如下面的例子是将定义直接写在表达式中或将整个表达式定义为符号变量。对于前者，合并同类项的运算可以自动进行，而对于后者，要用符号运算指令 collect 才能完成。对两种方法建立的两个表达式赋值以后相减，其结果为零，可见它们实际上是一样的。

```
    >> y = sym( 'x' ) + 2*sym( 'x' )
y =
    3 * x
>> y = sym( 'x+2*x' )
y =
    x + 2 * x
>> collect(y)
ans =
      3 * x
>> sin( 0:0.2:1 ) - subs( sin( sym( 'x' ) ), 0:0.2:1 )
ans =
      0    0    0    0   0   0
```

在实际计算中，两者各有不同的用途。符号运算可以用于公式推导和证明。下面证明平面旋转矩阵是一个正交矩阵。在平面上将坐标旋转角度 t，可以用变换矩阵 $\boldsymbol{G}$ 表示，相应的指令是

```
>> syms t;
>> G = [cos(t)  sin(t);  -sin(t)  cos(t)]
G =
   [ cos(t),   sin(t)]
   [-sin(t),   cos(t)]
```

连续两次变换的命令是

```
>> A = G * G
A =
   [ cos(t)^2-sin(t)^2,       2*cos(t)*sin(t)]
   [  -2*cos(t)*sin(t),     cos(t)^2-sin(t)^2]
```

再将它化简

```
>> A = simple(A)
A =
   [ cos(2*t),   sin(2*t)]
   [-sin(2*t),   cos(2*t)]
```

这就是连续两次转动的变换矩阵公式。下面再来证明变换矩阵 $\boldsymbol{G}$ 是一个正交矩阵，证明的方法是将 $\boldsymbol{G}$ 转置以后再与 $\boldsymbol{G}$ 相乘会得到单位矩阵：

```
>> I = G.' *G
I =
    [cos(t)^2+sin(t)^2,                    0]
    [                0,    cos(t)^2+sin(t)^2]
```

再化简得

```
>> I = simple(I)
I =
   [1,  0]
   [0,  1]
```

在 MATLAB 中有一个图形界面的函数计算器，调用的指令是 funtool 。用它可以对两个初等函数进行加、减、乘、除以及积分、微分、求反函数等运算。它就是用 MAPLE 的符号运算功能开发的。这个工具非常有趣地显示了符号计算与数值计算的差异。函数计算器的面板上有一个 demo 按钮，显示如何从 reset 键开始 (即从 x 开始)，经过九次点击面板上的不同按钮，构造出一个正弦函数 $\sin(x)$ 。函数计算器的设计者还在 help 键的说明中提出挑战，如果有人能

用更少的次数点击面板按钮就能产生一个正弦函数，请给他发封电子邮件。在 MATLAB 5.3 以后的版本新增加了函数的泰勒展开计算器，用指令 taylortool 调用，这对计算初等函数的泰勒展开十分方便。

用符号变量生成带参数的函数也很方便。例如要生成带参数 θ 的 x 的函数

$$FF = \cos^2(x^2) + \theta$$

并求出在 $x = 3$,$\theta = 2.1$ 时的函数值，然后求出函数的导数函数。做法是

```
>> GG = sym( 'cos( x^2 )^2 + theta' )
GG =
     cos(x^2)^2 + theta
>> subs( GG, {'x', 'theta'}, {3, 2.1})
ans =
      2.9302
>> diff( GG, 'x' )
ans =
      -4 * cos(x^2) * sin(x^2) * x
```

1.2.4 其它数据结构

下面介绍另外几种数据结构，如列阵 (array) 用以存储多维数组；数据网格 (meshgrid) 用以存储二元或三元的 “数据对”；基元列阵 (cell) 用以存储大小不同的矩阵甚至是空矩阵，结构数组 (struct) 可同时存入字符串和数据；最后介绍文本（即字符文字内容）的输入与存储。如果想要了解更多的数据表现形式，可以键入 help datatypes 进行查看。

1. 列阵

列阵就是多维数组。在使用 MATLAB 编程时，一个矩阵系列 $\boldsymbol{A}_k$ 就是一个三维列阵。第 k 个矩阵的第 (i, j) 个元素表示为 A(i, j, k) 。如果想象每一个矩阵都构成空间的一个层，则这三个指标分别表示元素所在的行、列和层。类似地，四维列阵有四个指标，第四个指标不妨称之为块，每一块都是一个三维列阵。第五维不妨称之为堆，每一堆中有若干块⋯⋯如此类推，很容易想象多维列阵在空间的排列顺序。

调用列阵的元素除了用行、列、层、块的下标之外，还可以用冒号算符，方法与在矩阵中使用的相似。

为了叙述方便，在此先约定几个术语。

维：维表示列阵的指标数，有三个指标叫三维列阵，有四个指标叫四维列阵。每一个指标都代表一个特定的维，有时候为了方便，也称之为一个特定的方向，用 dim 表示，并规定 dim 取 1， 2， 3， 4，⋯分别表示取列指标、行指标、层指标、块指标⋯⋯ dims 则表示由所有的 dim 组成的矢量。

维的元素数：每一个指标的取值范围就是它所代表的维的元素数。例如行的元素数是 20 表明它有 20 行，列的元素数是 5 表明它有 5 列，层的元素数是 3 表明它有 3 个矩阵，其余以此类推。列阵的总的元素数常表示成 m×n×l× ⋯的形式。

理解列阵元素的指标应遵循从大到小的原则，也就是要从后向前加以分析。以 A(4, 7, 5, 3) 为例，它表明 A 是四维列阵，调用的是其中第 3 块中的第 5 个矩阵中的第 7 列第 4 行的元素，而 A(:， :, 5, 3) 则是调用第 3 块第 5 个矩阵的全部元素。

由此可见，对于二维以上的列阵，维的元素数取 1 是无意义的。比如一个三维列阵的层的元素数为 1，它实际上就是一个矩阵，如果计算中不需要这个指标就可以把它取消。取消它们的指令是 squeeze，它会自动删除列阵中所有维的元素数为 1 的指标。 shiftdim 是移动列阵的各个指标，移动后它会自动删除维的元素数为 1 的指标。

生成列阵的方法与生成矩阵的方法相似，可以输入列阵的元素，包括整行、整列甚至整个矩阵；也可以利用 MATLAB 指令生成，如 randn， ones 和 zeros 都可以用于生成列阵。

指令 cat 可以用矩阵生成列阵，语句格式为

cat (dim, A1, A2, A3, ⋯)

其意义是沿着 dim 指定的方向将 A1， A2， A3，⋯ 组合成一个列阵。这里的 dim 取 1， 2， 3，⋯ 时分别表示列、行、层⋯⋯指令 cat 可以嵌套使用。例如下面的三个操作分别是将矩阵 $\boldsymbol{A}$ 和 $\boldsymbol{B}$ 按列、按行、按层组合。

```
>> A = [ 1 3; 5 7 ]
A =
    1     3
    5     7
>> B = [ 2 4; 6 8 ]
B =
    2     4
    6     8
>> cat( 1, A, B )
ans =
```



```
      1     3
      5     7
      2     4
      6     8
>> cat( 2, A, B )
ans =
      1     3     2     4
      5     7     6     8
>> D = cat( 3, A, B )
D(:,:,1) =
           1     3
           5     7
D(:,:,2) =
           2     4
           6     8
```

如果通过复制一个矩阵来生成列阵，则用指令 repmat(A， [m， n， p ⋯])。它将 A 在列、行、层⋯⋯的方向分别复制 m， n， p ⋯ 次。例如，下面的例子分别将 A 在列、行、层的方向分别复制 2 次、 3 次和 3 次。

```
>> repmat( A, [ 2, 3, 3 ] )
ans(:,:,1) =
             1     3     1     3     1     3
             5     7     5     7     5     7
             1     3     1     3     1     3
             5     7     5     7     5     7
ans(:,:,2) =
             1     3     1     3     1     3
             5     7     5     7     5     7
             1     3     1     3     1     3
             5     7     5     7     5     7
ans(:,:,3) =
             1     3     1     3     1     3
             5     7     5     7     5     7
             1     3     1     3     1     3
             5     7     5     7     5     7
```

对已有的列阵可以进行以下的变形操作：将所有的元素重排 reshape，将列阵的维数重排 permute 和恢复 ipermute，移动列阵的各维 shiftdim，删除列阵的维数为 1 的指标 squeeze 等。查看 help 很容易学会它们的用法。

对列阵使用 MATLAB 的指令仍然要注意指令的作用对象是标量、矢量，还是矩阵，不可混淆。如 sin 可以用于三维的列阵，而 eig 则只能用于矩阵，不能用于多维列阵，所以使用时只能对三维列阵中各层的矩阵分别应用。

2. 数据网格

平面上一个点的坐标要用一对数据 (x,y) 表示(不妨称之为二元"数据对")，要把平面上一个区域内所有的点都表示出来，就需要两个矩阵：矩阵 $\boldsymbol{X}$ 和矩阵 $\boldsymbol{Y}$。所以 $\boldsymbol{X},\boldsymbol{Y}$ 就是平面上的数据网格。在这两个矩阵的对应位置上，每取一对数据都可以表示平面上的一个点。类似地，空间的点要用三元"数据对" (x,y,z) 来表示，整个空间的点要用三个三维列阵 $\boldsymbol{X},\boldsymbol{Y},\boldsymbol{Z}$ 才能表示，所以 $\boldsymbol{X},\boldsymbol{Y},\boldsymbol{Z}$ 就是空间的数据网格。以此类推，n 维空间的点要用 n 个 n 维列阵表示。这种数据结构既不是矩阵，也不是多维数组，我们以后称它为数据网格。画三维图形或求出某个区域内二元函数的值要用二维的数据网格，而求空间区域中三元函数的值则要用到三维的数据网格。MATLAB 构造数据网格的指令是 meshgrid 和 ngrid。

(1) meshgrid　　生成二维或三维数据网格

调用的语句格式为

[X, Y] = meshgrid(x, y)

[X, Y] = meshgrid(x)

[X, Y, Z] = meshgrid(x, y, z)

其中输入量 x, y, z 是三个矢量，表示数据网格在三个方向上的取值分布。如果只输入一个矢量，则默认其余的矢量也取同样的值。

输出量 [X, Y] 是所要的二维数据网格，其中矩阵 X 的每一行都是矢量 x，矩阵 Y 的每一列都是矢量 y。[X, Y, Z] 是输出的三维数据网格。

(2) ndgrid　　生成 n 维数据网格

调用的语句格式为

[X1, X2, X3, ⋯] = ndgrid(x1, x2, x3, ⋯)

[X1, X2, X3, ⋯] = ndgrid(x)

各项符号含义基本与 meshgrid 相同。在用于二维和三维网格时，它相当于将 meshgrid 输入和输出的前两行交换位置以后所得的结果，即

```
>> [X, Y, Z] = meshgrid( x, y, z )
```

与

```
>> [X, Y, Z] = ndgrid( x, y, z )
```

的结果是不一样的，而

```
>> [X, Y, Z] = meshgrid( x, y, z )
```

与

```
>> [Y, X, Z] = ndgrid( y, x, z )
```

的结果是一样的。

下面是由矢量 $\boldsymbol{x}, \boldsymbol{y}$ 生成的二维数据网格 $\boldsymbol{X}, \boldsymbol{Y}$

```
>> x = [ 1 3 5 ];     y = [ 2 4 6 ];
>> [X Y] = meshgrid( x, y )
X =
     1     3     5
     1     3     5
     1     3     5
Y =
     2     2     2
     4     4     4
     6     6     6
```

由 $\boldsymbol{X}, \boldsymbol{Y}$ 在对应位置上每取一个元素，就表示平面上一个点的坐标，所以 $\boldsymbol{X}, \boldsymbol{Y}$ 共代表了九个点的坐标，分别是

```
(1, 2)   (3, 2)   (5, 2)
(1, 4)   (3, 4)   (5, 4)
(1, 6)   (3, 6)   (5, 6)
```

下面的例子表明由矢量 $\boldsymbol{x}, \boldsymbol{y}$ 生成的函数 $\boldsymbol{z}$ 仍是矢量，矢量 $\boldsymbol{z}$ 只有三个值，而数据网格 $\boldsymbol{X}, \boldsymbol{Y}$ 生成的函数 $\boldsymbol{Z}$ 则是矩阵，矩阵 $\boldsymbol{Z}$ 有九个值，这九个值与平面上的九个点相对应。所以利用数据网格 $\boldsymbol{X}, \boldsymbol{Y}$ 才能画出空间的曲面。

```
>> z = x.^2 + 2*y
z =
     5    17    37
>> Z = X.^2 + 2*Y
Z =
     5    13    29
     9    17    33
    13    21    37
```

三维数据网格的数值结构与二维数据网格相似，但是要用三个三维列阵表示。下面这个例子是用三个矢量 $\boldsymbol{x}, \boldsymbol{y}, \boldsymbol{z}$ 组成了三维的数据网格 $\boldsymbol{X}, \boldsymbol{Y}, \boldsymbol{Z}$ 。

```
>> x = [ 1 4 7 ];    y = [ 2 5 8 ];    z = [ 3 6 9 ];
>> [ X, Y, Z ] = meshgrid( x, y, z )
X(:,:,1) =
           1     4     7
           1     4     7
           1     4     7
X(:,:,2) =
           1     4     7
           1     4     7
           1     4     7
X(:,:,3) =
           1     4     7
           1     4     7
           1     4     7
Y(:,:,1) =
           2     2     2
           5     5     5
           8     8     8
Y(:,:,2) =
           2     2     2
           5     5     5
           8     8     8
Y(:,:,3) =
           2     2     2
           5     5     5
           8     8     8
Z(:,:,1) =
           3     3     3
           3     3     3
           3     3     3
Z(:,:,2) =
           6     6     6
           6     6     6
           6     6     6
Z(:,:,3) =
```



```
        9    9    9
        9    9    9
        9    9    9
```

这里的 $\boldsymbol{X}$, $\boldsymbol{Y}$, $\boldsymbol{Z}$ 都是三维的列阵，这个数据网格表示了空间的 27 个点的坐标，当每个列阵的第三个指标（层指标）都取 1 时，表示的是 $z = 3$ 的各个点的坐标

```
( 1, 2, 3 )   ( 4, 2, 3 )   ( 7, 2, 3 )
( 1, 5, 3 )   ( 4, 5, 3 )   ( 7, 5, 3 )
( 1, 8, 3 )   ( 4, 8, 3 )   ( 7, 8, 3 )
```

3. 基元列阵

基元列阵 (cell array) 可以将不同类型的数据按照与矩阵形式相似的结构组织起来加以应用。

例如下面就建立一个基元列阵

```
>> G{1,1}=3;
>> G{1,2}=[1,2; 3,5];
>> G{2,1}='good';
>> G{2,2}='sin(x)';
>> G
G =
      [3]      [2×2  double]
    'good'         'sin(x)'
```

其中有一个元素是矩阵，所以只给出了它们的维数和数据精度。如要查看它的内容，可以将它们调出来，方法与调用矩阵元素的方法相同。如调用其中的矩阵元素，操作是

```
>> G{1,2}
ans =
      1     2
      3     5
```

基元列阵用途广泛，读者可以参阅本书的参考文献。

4. 结构数组

结构数组是比基元列阵更复杂的数据结构，它不仅可以将不同类型的数据组织在一起，还可以为它们赋予名称。所以结构数组是一种有域名的数据，它类似于关系数据库中数据的结构。建立结构数组的语句是

s = struct('field1', values1, 'field2', values2, ⋯)
其中 field1， field2 是域名， values1， values2 是相应的数据。数据元素要与域名相对应。

例如，下面的指令产生一个结构数组 s，域名是 type， color， x，输入了两组数据，可用 s(1)， s(2) 查看。域名是文本型的字符串，用引号括起来；数据是基元列阵，用花括号括起来，其中的文本型的字符串还是要用引号，而标量数据则不用引号。

```
>> s = struct('type', {'big', 'little'}, 'color', {'red'}, 'x', {3 4})
s =
    1×2 struct array with fields:
    type
    color
    x
>> s(1)
ans =
       type:  'big'
      color:  'red'
          x:  3
>> s(2)
ans =
       type:  'little'
      color:  'red'
          x:  4
```

对结构数组操作的指令还有 fieldnames, getfield, rmfield, setfield。在解非线性方程或解常微分方程时，为了说明解方程的条件就要用结构数组。结构数组也是一种用途广泛的数据结构，读者可以参看本书的参考文献。

5. 字符和文本

输入文本的方法是用单引号，它输入的是 1×n 的列阵。如

```
>> s = 'Hello'
```

是一个 1×5 的列阵。在 MATLAB 内部它是作为数来保存，但不是浮点数。如下的操作是将字符列阵转换为浮点数列阵，每个数代表相应字符的 ASCII 码。

```
>> a = double(s)
a =
    72    101    108    108    111
```

而反向操作

```
>> s = char(a)
```

把数再变成字符。指令 char 是显示编码表中的字符，当编号是 32 到 127 时，显示的是 ASCII 码表中的字符。

可以用方括号把字符合并为一个大的字符串，如水平合并和垂直合并的操作分别是

```
>> h = [ s, ' world' ]
h =
    Hello world
>> v = [ s; 'world' ]
v =
   Hello
   world
```

1.3 编程

本节介绍 MATLAB 的编程语言、程序文件的编辑调试、数据的输入输出。

1.3.1 程序文件的编辑与调试

MATLAB 不仅是一个交互式计算工具，也是一种效率极高的编程语言。只有掌握了编程，才能真正开始下面的学习。学习编程的必经之路是多多阅读好的程序，如 MATLAB 的许多函数和程序都可以直接调出来阅读，通过阅读这些程序就能掌握许多文字难以说清楚的巧妙的编程技巧。

MATLAB 的程序文件都是以 .m 为扩展名，所以称为 M 文件。在 MATLAB 中带有一个编辑器可以编辑 M 文件。 MATLAB 中的许多指令和全部的工具箱文件都是 M 文件。在 M 文件中，凡是说明性的文字都用 %开头。

1. 编辑 M 文件

现在来编辑如下内容的程序文件，并命名它为 mag.m 。

```
%% program  mag.m
A = [16.0     3.0      2.0    13.0
      5.0    10.0     11.0     8.0
      9.0     6.0      7.0    12.0
      4.0    15.0     14.0     1.0]
```

```
B=A′
C=eig(A)
```

编辑过程如下：

①依次逐层点击 MATLAB 指令窗的菜单 File\New\Mfile ，打开图 1.2 所示的程序编辑器窗口

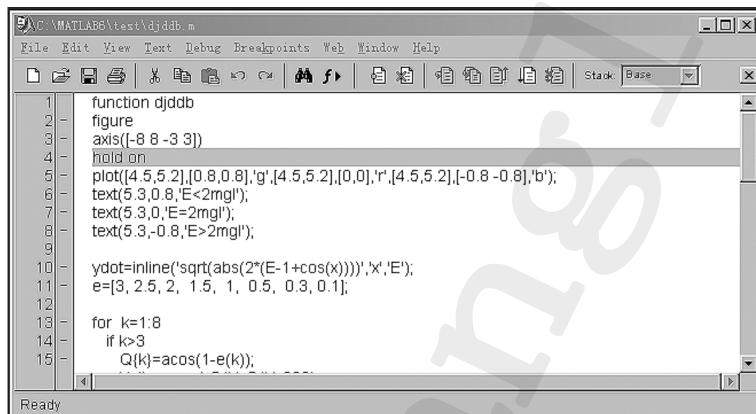


图 1.2 MATLAB 的程序编辑器窗口

这个编辑器兼有编辑与调试两种功能。在窗口中可以输入该程序的内容并进行编辑，注意在程序文件中，各指令前面没有 << 符号。

②程序编好以后，进行存盘。方法是依次逐层点击菜单 File\Save As, 就会打开一个对话框，选择存盘的目录，并在文件名一栏中输入 mag.m ，然后选择保存。

在指令窗键入 mag ， MATLAB 就会运行这个程序。程序运行的结果是建立了矩阵 $\boldsymbol{A}$ ，同时还对 $\boldsymbol{A}$ 进行了转置和求本征值的运算。

编辑一个已有的 M 文件，可以利用 File 菜单下的 Open 子菜单，也可以在指令窗直接键入

```
>> edit 文件名
```

MATLAB 可以同时编辑多个文件，所有打开的文件名都以标签的形式显示在窗口的下面。

M 文件的命名规则与一般的文件命名规则相同，文件名要以字母开头，不要以数字开头，最好也不要用 MATLAB 默认的文件名 untitled 。

2. 编辑器的功能

在 MATLAB 6.0 的程序编辑器窗口中的操作界面相对于以前的版本有了显著的变化，它增加了以下几项功能

①窗口的左边可以显示行号。

②程序中的汉字注释随时都能正确显示。而在以前的版本中，如要在程序中正确显示汉字注释，必须先打开编辑器窗口，再将文字输入状态从英文转换到中文，最后打开程序文件，这样汉字才不会显示乱码。

③增加了一个菜单命令 Text，其中包括以下命令

Comment　注释选定的行

Uncomment　取消注释

Decrease Indent　减少各行缩进的程度

Increase Indent　增加各行缩进的程度

Balance Delimiters　调整分界符

Smart Indent　将选定的各行对齐

Evaluate Selection　运行选定的内容

3. 程序调试

在编辑器窗口调试程序十分方便，这些功能主要在两个菜单下。在菜单 Breakpoint 下的功能有

Set/Clear Breakpoint　设置或清除中断点

Clear All Breakpoint　清除全部中断点

Stop If Error　遇到错误中止运行

Stop If Warning　遇到警告中止运行

Stop If NaN Or Inf　遇到 NaN 或 Inf 中止运行

在 Debug 菜单下的功能有

Step　逐步运行

Step In　进入干预状态

Step Out　退出干预状态

Continue　连续运行直到一个中断点

Save and Run　保存程序并运行

Run　运行程序

Go Until Cursor　运行到光标位置

Exit Debug Mode　退出程序调试

其中 Continue， Save and Run， Run 是在同一个位置，根据程序状态自动显示的不同命令。如果程序编辑后尚未保存，则显示 Save and Run；如果是打开一个已有的程序，则显示的是 Run；如果程序处在运行状态，则显示的是 Continue。当程序运行到某一行时，如果该行中有 MATLAB 的指令，可以选择 Step In 以打开与指令有关的 M 文件，查看程序在其中的运行情况。退出这种状态可以选择 Step Out。

上述这些指令在窗口中都有相应的图标可以使用。在窗口左边的行号旁边有一个竖条，用鼠标双击其中的小横线，也能设置中断点。

调试程序的含义有两方面，一方面是改正程序中的语法错误，如写错了函数名，少写了半个括号等， MATLAB 能够自动查出大多数这类错误；另一方面是改正程序编写错误，程序编写错误会造成程序不能运行下去。

要调试一个编好的程序，常常是在程序中设置中断点来分段运行程序，同时也可设置停止运行的条件，如遇到错误就中止运行等。设置中断点以后，在程序的相应位置会出现一个红圆点。然后开始运行程序，当程序运行到中断点的位置，便会停止下来，等待输入指令，同时在该位置会出现一个绿色的箭头。根据需要可用不同的指令去控制程序的运行，如 Step, Continue 等。绿色的箭头随时都会指示程序运行所到的位置。当程序运行出错时，会将错误信息显示在指令窗中，根据这种信息可以修改程序。如此进行，直至程序能完全通过为止。

为了提高程序的运行速度，必须做到避免不必要的和重复的计算，采用最节省时间的算法。 MATLAB 提供了一种评价程序的工具 Profile 。它可以记录程序每一步运行的时间，并将结果用图形或报告显示，使你能够找到问题的所在，以便于改进。如评价程序 ch3.m ，做法如下

① 键入指令 profile on　　启动 profile
② 键入程序 ch3　　运行程序 ch3
③ 键入指令 profile report　　生成文字评价报告
④ 键入指令 profile plot　　生成图形评价报告

程序在调试好以后，就可以使用了。但这并不意味着这个程序一定是正确的，也就是编程工作到此还不能算结束。因为目前只能说明这个程序符合编程语法的要求，而一个程序的正确性不仅与算法有关，更重要的是还与构造算法的物理模型、物理思想和物理概念有关。为了检验程序的正确与否，往往要选用以下方法

① 根据物理模型，对可能得到的结果进行一些定性分析，用以预测计算结果。在可用解析方法求出一些简单解时，可把解析结果与程序计算的结果进行对比。

② 改变程序中的参数重新进行计算，分析比较所得的结果，看它们表现出的规律性是否能互相印证。

③ 如有可能，对物理模型再设计一个不同的算法，编辑新的程序进行计算，以检验旧程序。

4. 设置搜索路径

自编的程序通常是存在 MATLAB 目录下的 work 文件夹，这是 MATLAB 提供给用户存储文件的文件夹，只要启动 MATLAB 就可以调用其中的程序。如果

将自己的目录加入到 MATLAB 的搜索路径中去，那么程序在存入自己的目录中以后也能正常调用。加入的方法是，双击 File 菜单下的 Set Path 命令，会打开一个路径设置窗口 (Set Path) 如图 1.3 所示。

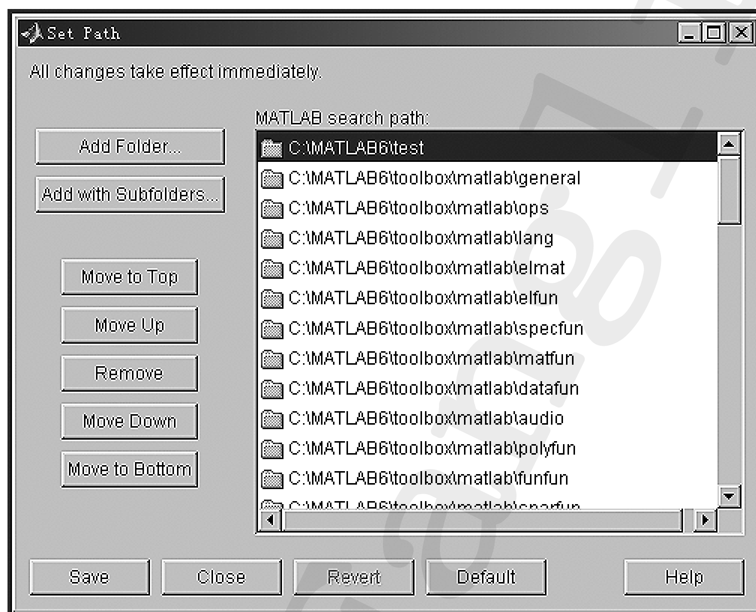


图 1.3 MATLAB 的路径设置窗口

再按照窗口中各个按钮上的指示进行操作即可。各个按钮的含义为

Add Folder　　把指定的目录加入搜索路径
Add with subfolders　　把指定目录及其子目录加入搜索路径
Move to Top　　将选定的目录移到最上面
Move to Bottom　　将选定的目录移到最下面
Move Up　　将选定的目录向上移动
Move Down　　选定的目录向下移动
Remove　　删除选定的目录
Save　　保存新加入的目录
Revert　　取消本次操作
Default　　恢复机器原来的默认设置

如果是 MATLAB 6.0 以前的版本，这个窗口操作界面会有所不同，但功能相似。

1.3.2 指令类文件和函数类文件

M 文件分指令类文件 (script files) 和函数类文件 (function files) 。指令类文件比较简单，它是将在指令窗下可以执行的一些函数和指令按照执行的顺序集中写在一个文件里，一起执行，效果类似于 DOS 下的批处理文件。它没有输入参数和输出参数，可以使用指令窗中的变量。由它建立的变量在文件执行完成以后也会保留在指令窗的工作内存中。在 1.3.1 节编写的程序 mag.m 就是一个指令类文件，程序中的命令都可以在指令窗中执行，编成程序以后就可以集中执行，而且可以反复使用。

1. 函数文件的结构

函数文件的编辑比较复杂，用途也更多。如后面为解微分方程而编写的 odefile 就是一种特殊格式的函数文件。下面重点介绍函数文件。

函数文件的结构可分以下几部分。

(1) 函数定义行

这是函数文件的第一行，其格式为

function [输出变量 Q1, Q2, ···] = 函数文件名 (输入变量 P1, P2, ···)

这里的函数文件名就是存储这个文件时所使用的名称，最好不要用其它名称来代替。尤其是在后面讲到的子函数文件中，如果使用其它名称，会发生子函数不能调用的错误。如果不使用输入和输出变量，也可以只写函数文件名，但开头的关键字 function 不可缺少。 P1， P2， Q1， Q2，··· 必须是函数文件中使用的变量名。

(2) H1 行

帮助文本的第一行。简单说明函数的基本功能，是指令 lookfor 查找的范围。

(3) 函数帮助文本

帮助文本的全文。详细介绍函数的功能和用法，是指令 help 查找的内容。

(4) 函数体

函数文件中的程序。包括流程控制、交互输入输出、计算、赋值、注释，也可以调用函数文件和指令类文件。

(5) 注释

为了帮助理解程序，可以在程序的任何位置加入注释文字，但必须用 %开头。在同一行内，所有在 %后面的内容都不会执行。

2. 函数文件的调用

函数文件的调用格式为

[输出变量 Q1, Q2, ···] = 函数文件名 (输入变量 P1, P2, ···)

参数要保持与函数文件相同的顺序，同时不能多于函数文件中的参数数目。函数文件中的变量默认为局部变量，只能在本函数文件中使用。只有将它设置成全局变量以后才能被其它函数文件使用或在指令窗使用。与函数文件交换数据一般是通过输入参数和输出参数。所以调用函数文件实际上就是通过输入参数给函数文件中变量 P1, P2, ··· 赋值，通过函数运算以后，返回变量 Q1, Q2, ··· 的值。由此可见，函数文件可以完成指令类文件的功能，但指令类文件却不具有函数文件的功能。

函数文件可以调用自己，称为递归调用。这样调用必须确保调用过程能够中止，否则，会陷入死循环。 1.4.7 节中的例 3 即小球弹跳的例子就是采用了递归调用。

用函数文件建立函数是建立函数的第三种方法。下面是两个不同的函数文件，文件名分别为 test1.m 和 test2.m ，它们建立了两个不同的函数，在 1.2 节曾用指令 inline 来建立这两个函数。文件中采用数组算符。

```
%% program1  test1.m
function FFF = test1(x,theta);
FFF = cos(x.^2).^2 + theta;

%%program2   test2.m
function ff = test2(v,c)
x = v(:,1);  y = v(:,2);  z = v(:,3);
ff = -x.^2 + y.^2 - z.^2 + 6 + c
```

将它们分别存盘以后，可按照 test1(x,theta) 和 test2(v,c) 格式调用它们。比如说，要计算

$$x = 1,\ y = 0.3,\ z = -2,\ c = -4$$

时和

$$\boldsymbol{X} = [1,\ 3,\ 6],\ \boldsymbol{Y} = [2,\ 2,\ 7],\ \boldsymbol{Z} = [3,\ 1,\ 8],\ c = 3$$

时 ff 的值，可键入

```
>> test2( [1, 0.3, -2], -4 )
ans =
      -2.9100
>> test2( [1, 2, 3; 3, 2, 1; 6, 7, 8], 3 )
ff =
     3
     3
     -42
```



3. 子函数文件

在同一个函数文件中可以建立多个函数，第一个函数叫主函数，其余的叫子函数。子函数只能被同一个文件中的主函数和其它子函数调用。这个功能很像 C 语言编程中的主程序与子程序的关系，这在编程中会带来很多的方便。主函数名就是函数文件名。应用子函数的实例可参见 1.4.7 节的例 3 。

4. 全局变量

如果几个函数文件要共用一个变量，那么要在这些函数文件中都定义这个变量是全局变量。如果在指令窗中也要使用这个变量，就还要在指令窗中定义这个变量为全局变量。必须在使用函数之前定义全局变量。例如，建立文件 falling.m

```
%program falling.m
function h = falling(t)
global GRAVITY
h = 1/2*GRAVITY*t.^2;
```

然后再在指令窗中依次键入

```
>> global GRAVITY
>> GRAVITY = 10;
>> y = falling((0:.1:5)');
```

这两个关于全局变量的表述使得在指令窗输入的 GRAVITY 的值能在函数内部使用。在指令窗输入不同的 GRAVITY 的值，比如键入

```
>>GRAVITY = 9.8
```

就能得出新解而不必去改变程序中 GRAVITY 的值。

1.3.3 流程控制

在 M 文件中，不但可以按顺序执行指令，也可以使用循环结构和选择结构来控制流程，还可以在程序的运行中，接受键盘输入的指令。这种 M 文件与 FORTRAN 和 C 语言中常说的程序已经很相似了。下面介绍控制流程的语句。

1. 循环结构

MATLAB 中有两种循环方式。一种是 for 循环，用于可以确定循环次数的循环，另一种是 while 循环，用于不能确定循环次数的循环。

(1) for 循环的语句格式是

for　循环变量 = 起始值 : 步长 : 终止值
　　循环体

end

步长的默认值是 1, 可以是任意的正实数或负实数。当步长为正数时，表示循环变量大于终止值时循环停止；当其为负数时，表示循环变量小于终止值时循环停止。 for 循环可以嵌套使用。 for 循环的应用参看本节例 1 。

(2) while 循环的语句格式是

while　　表达式

**　　　　循环体**

end

当表达式为真时，执行循环体，执行后再判断表达式是否为真，是真则执行循环体，否则退出循环体。例如要找出小于 10^{10} 的最大阶乘，用的程序是

```
n=1
while prod(1:n)<1e10
   n=n+1
end
```

得到的结果是 14 即 14 ！是小于 10^{10} 的最大阶乘。

在其它语言中要使用 for 或 do 循环的地方， MATLAB 可以用矢量或矩阵操作。如建立一个对数表，在其它的语言中的程序是

```
x = 0;
for k = 1:1001
   y(k) = log10(x)
   x = x + .01;
end
```

在 MATLAB 中的程序应该是

```
x = 0:.01:10;
y = log10(x)
```

这称之为矢量化编程。在 MATLAB 中，只有确实不能矢量化的程序才有必要使用 for 循环，否则会大大减慢程序运行的速度。很多初学者往往觉得， MATLAB 虽然方便，但计算速度似乎不快，其实就是因为在程序中不正确地运用了 for 循环之类的语句。矢量化编程优化了其它语言中的 for 语句，所以运算速度更快。在指令窗键入 xplang, 可以调出一个演示程序，其中有一句话很有代表性，

“ Life is too short to spend writing DO loops ⋯ ”

这是 MATLAB 编程者之间常说的一句话,“宝贵而短暂的生命可不能浪费在做 DO 循环上⋯⋯”，它充分表达了编程者对 MATLAB 运算速度的自信心和自豪感。

如果一段程序不能矢量化，可以预先对一个矢量或矩阵设定初值，那样可以加快 for 循环。下面的例子是用函数 zeros 对 for 循环建立的矢量设定初值。

它明显地加快了 for 循环的速度。

```
r = zeros(32,1);
for n = 1:32
   r(n) = rank(magic(n))
end
```

如果不设定初值， MATLAB 的编译器增大矢量 r 的方法是每循环一次只增加一个元素。预先设定初值取消了这个步骤，从而加快了执行速度。

2. 分支结构

MATLAB 中的分支结构有 if 结构和 switch 结构。前者用逻辑表达式作判别式，后者用字符串或数值作判别式。

(1) if 结构的语句格式为

```
if      逻辑表达式 1
        语句体 1
elseif     逻辑表达式 2
           语句体 2
else
        语句体 3
end
```

上述结构可以简化为 if-end 或 if-else-end 结构。在语句执行时，如果逻辑表达式 1 为真，则执行语句 1，如果为假，则判断逻辑表达式 2 是否为真，如果为真，则执行语句 2，否则向下执行。 if 分支结构也可以嵌套使用。

(2) switch 结构的语句格式为

```
switch      表达式（标量或字符串）
case     值 1
    语句 1
case     值 2
    语句 2
    … …
otherwise
    语句 3
end
```

这种结构是在几种情形下选择其中的一种来执行。如果表达式与值 1 相符，则执行语句 1；如果表达式与值 2 相符，则执行语句 2；以此类推，如果都不相符就执行语句 3 。然后才会执行 end 以后的程序。在 switch 结构中也可以

不包括 otherwise 语句。在前面多次引用过的 1.4.7 节例 3 的函数文件 ballode, 就运用了 switch 结构来判断参量 flag 的值。

3. 其它指令

break　用于 for, while, if 语句的中止，如果循环是嵌套的，则只从最内的一层退出。

pause　使程序运行暂停，按任意键恢复运行，而 pause(n) 则是暂停 n 秒。

disp(′ ··· ′)　在屏幕上显示引号中的内容。

input　将用户从键盘输入的数值、字符串或表达式赋予指定的变量。例如键入

```
>> a=input ( 'Please input a number:' )
```

运行以后，在屏幕上将会显示引号内的文字，然后可从键盘输入数字、表达式或带单引号的字符串，回车以后它们被赋予变量 a 。如要输入字符串，则格式为

```
>> a=input( 'Please input a string:' , 's' )
```

运行以后，在屏幕上也会显示引号内的文字，然后也可从键盘输入带单引号的数字、表达式或带单引号的字符串，回车以后它们是作为字符串被赋予变量 a 。

keyboard　暂时中止程序的运行，等待键盘输入的指令，在执行完输入的指令以后，只要键入 return, 又可以恢复程序的运行。常用于调试程序或在程序运行中修改变量的值。

下面的程序中都用了 for 语句控制流程。程序中的 plot(x,y) 是用函数值 y 对变量 x 画二维曲线，而 subplot 是画分区图的指令。

例 1　著名的生态学模型 —— Logistic 模型是一个一维非线性迭代方程：

$$x_{n+1} = \mu(x_n - x_n^2)$$

将参数 μ 限制在 $(0, 4)$, x 就在 $(0, 1)$ 变化。计算中令参数 μ 值从 2.6 逐渐增大，对 x 进行迭代，在达到一定的迭代次数后，系统的稳定状态会从一个变成两个，又从两个变成四个，··· ··· 最终呈现混沌现象。

在编程计算时，通过试验，发现在迭代 150 次后，系统可达到稳定。因此在程序中采用了两个循环，第一个循环迭代 150 次，但不画图；第二个循环用第一个循环的最终结果作初始值进行计算，并且画图。第二个循环的迭代次数是通过试验决定的。对不同的 μ 值不再采用 for 循环，而是采用矢量表示，以加快计算速度。实际程序 bug.m 如下（图 1.4 是程序运行时所画出的周期分岔图）：

```
%%sprogram    bug.m
x=0.6;        %%x 的初始值
u=2.6:0.001:4;        %%参数的取值范围
for j=1:150        %%此循环让系统达到稳定态
```

```
    x=u.*(x-x.^2);         %%对 x 进行迭代
end
for i=1:100        %%此循环输出分岔图
    x=u.*(x-x.^2);
    plot(u,x,'r.','marker','.','markersize',1)
    hold on;
end
```

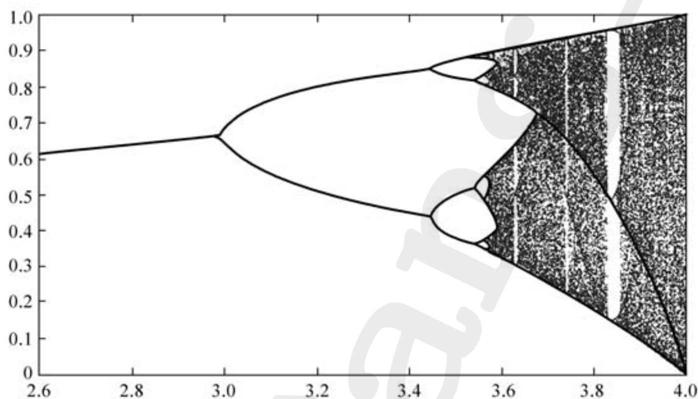


图 1.4　Logistic 模型的周期分岔图

例 2　雪花的分形图。将等边三角形的一条边三等分，将中间的一段去掉，代之以一个更小的等边三角形的两条边，依次做下去，将得到图 1.5 所示的雪花图。

在程序 snow.m 中用复数的实部和虚部分别表示点的坐标 x,y 。 plot 是作图指令，它对复数作图相当于用复数的实部和虚部分别表示曲线的 x,y 坐标。因此只要计算出各点的坐标，依次连接起来，就是所要的结果。

程序中的第一句是列出第一个分区图中 a,b,c,d 共 4 个点的坐标，其中 a,d 两个点是重合的，然后画出一个等边三角形。接下来对 k 的循环画出了后面的 5 个分区图。而对 jj 的循环，是对每条边计算新图形中 4 个点的坐标。以第二个分区图为例， jj 共有 3 次循环，分别计算了 $(a,b,c,d),(e,f,g,h),(i,j,k,l)$ 点的坐标，而 m 的坐标则等于 a 点的坐标。程序 snow.m 如下：

```
%%program snow.m
new=[0.5+(sqrt(3)/2*i),-0.5+(sqrt(3)/2*i),0,0.5+(sqrt(3)/2*i)];
     %%给出分区图 1(图 1.5(a)) 中 a,b,c,d 点的坐标
subplot(2,3,1);  plot(new),  axis equal       %%画出分区图 1
```

```
for k=1:5;      %%画分区图 2~6(图 1.5(b)~(f))
   old=new;      %%保存原有各点的坐标
   n=length(old)-1;     %%计算点的数目
   for jj=0:n-1;      %% 对每条边计算 4 个新的点的坐标
      diff=(old(jj+2)-old(jj+1))/3;
      new(4*jj+1)=old(jj+1);
      new(4*jj+2)=old(jj+1)+diff;
      new(4*jj+3)=new(4*jj+2)+diff*((1-sqrt(3)*i)/2);
      new(4*jj+4)=old(jj+1)+2*diff;
   end
new(4*n+1)=old(n+1);      %%最后一个点的坐标与第一个点的坐标相同
subplot(2,3,k+1);
plot(new)
axis equal;     %%画出分区图
end
```

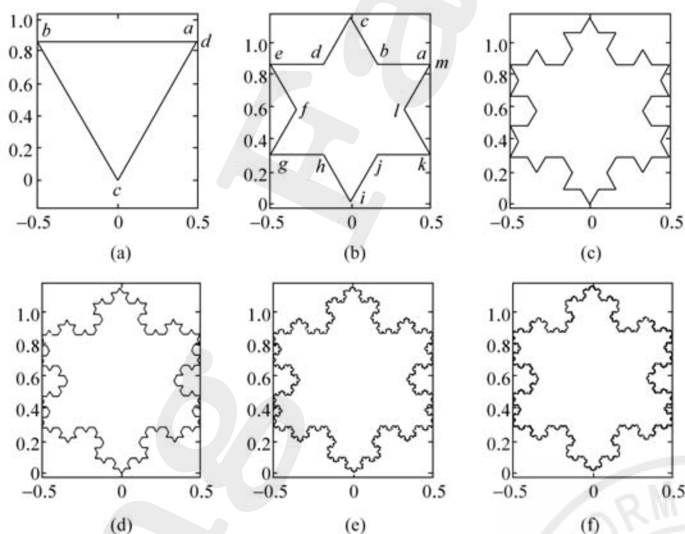


图 1.5　雪花形状的分形图

例 3　树形分形图。将一条线段三等分，在等分点上各画一条长度为原线段长度的三分之一的线段，该线段与原线段成固定的夹角。依次做下去，就会得到图 1.6 所示的树形图。

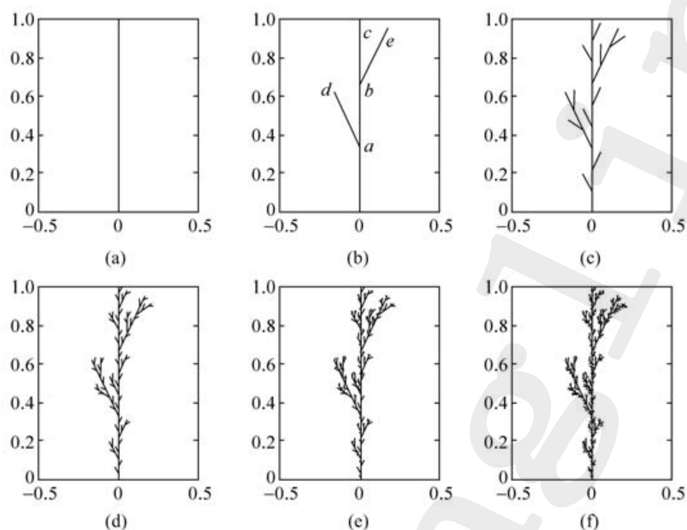


图 1.6 树形分形图

实际的程序 tree.m 如下 (程序中的注解只是对分区图 1, 2 (图 1.6(a)(b)) 的解释, 对于以后的图形可以类推):

```
%% program  tree.m
theta=pi/6;      %%设置夹角
u =[0,0;0,1];
subplot(2,3,1)
plot(u(:,1),u(:,2))      %%画分区图 1 ( 图 1.6(a)) 中的直线
axis([-0.5,0.5,0,1])
for n=1:5       %%每次循环画出一个分区图
   uuu=[ ];      %%建立一个空的矢量, 存放点的坐标数据
   for I=0:(length(u)/2-1)      %%对每一条线段计算新的点的位置
      p1=(u(2*I+1,:)*2+u(2*I+2,:))/3 ;
      p2=(u(2*I+1,:)+u(2*I+2,:)*2)/3;
      lp=[cos(theta),-sin(theta);sin(theta),cos(theta)]*...
          [(u(2*I+2,1)-u(2*I+1,1));u(2*I+2,2)-u(2*I+1,2)]/3 ;
      lp=p1+lp';      %% d 点的位置坐标
      rp=[cos(theta),sin(theta);-sin(theta),cos(theta)]*...
          [(u(2*I+2,1)-u(2*I+1,1));u(2*I+2,2)-u(2*I+1,2)]/3;
      rp=p2+rp';      %% e 点的位置坐标
      uu=[u(2*I+1,:);p1;p1;lp;p1;p2;p2;rp;p2;u(2*I+2,:)];
```

```
    %%按照 (0,a,a,d,a,b,b,e,b,c) 顺序排列数据，以供作图用
      uuu=[uuu;uu];      %%将新的点的坐标存入矢量中
    end
    u=[uuu];
    subplot(2,3,n+1)     %%画分区图
    plot(u(:,1),u(:,2))
    axis([-0.5,0.5,0,1])
end
```

1.3.4 数据输入与输出

MATLAB 与外部环境交换数据，常用 save 和 load 这两个指令，其用法为

1. save　将内存中的变量存入文件

其语句格式有以下几种：

save fname　将工作内存中的变量存入二进制文件 filename.mat 中。

save fname X Y Z　存储变量 X，Y，Z。可以用通配符 ∗。

save fname X Y Z -ascii　文件为 8 位的 ASCII 文件。

save fname X Y Z -ascii -double　文件为 16 位的 ASCII 文件。

save fname X Y Z -ascii -double -tabs　列表式的文件。

save fname X Y Z -append　将新的变量加入到已有的文件中。

2. load　将文件数据读入内存

其语句格式有以下几种：

load filename　从文件 filename.mat 中读入数据。必须指明有关的路径。

load filename x, y, z, ⋯　只读入指定的变量 x, y, z，⋯ 变量名也可以用通配符。

load filename.ext　读入 ASCII 码的分行形式的数据，文件中可以包含以 %开头的 MATLAB 的命令，变量名与文件同名。

load filename -ascii(or -mat)　按 ASCII 码或二进制读入文件数据。

指令 load 也可读入包含数值数据的文本文件。文本文件应该列成数据的表，同行中的各列用空格分开，每行中有相等的元素。例如用任何编辑器编制如下数据表

```
16.0     3.0     2.0    13.0
 5.0    10.0    11.0     8.0
 9.0     6.0     7.0    12.0
```

并将它存入 d 盘的 mag.dat 文件。要将它读入工作空间，可键入命令

```
>> load d:\mag.dat
```

就会将数据文件读入内存并建立变量 mag, 它代表文件中的矩阵。

在附录 A 中有更多的有关文件操作的指令可供读者参考。

1.4 常用的计算指令

物理中常用的计算指令有两种，一种是对数据的操作指令，如求差分、微分、积分等；另一种是对函数操作的指令，如求根、解微分方程等，在 MATLAB 中称之为 function functions ，它们的作用对象是 M 文件建立的函数或者是用 inline 指令建立的函数。这些指令的差别很大，例如查找矩阵元素的最大值和求一个函数的最大值就是不同的指令。不断将各种指令进行对比，会有助于加深对它们的理解。

1.4.1 插值与曲线拟合

MATLAB 的插值计算功能非常强大，除了有一些指令可直接调用之外，还有一个专门的工具箱 Spline Toolbox, 可以完成许多复杂的插值计算，在指令窗口键入

```
>> help splines
```

可以查看。本节只介绍几条简单的指令。

1. 插值法

计算插值的指令有 interp1, interp1q 和 interp2 等，分别介绍如下：

(1) interp1　一维插值

调用的语句格式有三种：

yi=interp1(x, y, xi)

yi=interp1(y, xi)

yi=interp1(x, y, xi, 'method')

在这里， (x, y) 是已知的样点构成的数据组，其中 x 是一维数组， y 是对应的函数值。 xi 是新的插值节点组成的一维数组， yi 是求出的相应的插值。

要求 x 是单调增加的，但可以是不等间距的。调用时如果未给出 x, 则默认 x = 1:n , n 为 y 的长度即元素的个数。 y 可以是矢量或矩阵。如果是矩阵，则对矩阵 y 中的每一列进行插值，得出的结果 yi 是 n×m 矩阵， n 是 xi 的长度， m 是 y 的行数。超出这个范围的值一律返回 NaN 。

method 是所用的插值算法，默认为线性算法，用户可选用以下算法中的一种。如果 x 是等间距的，使用 nearest, linear, cubic 算法的运算速度要快一些。

'nearest'	线性最近项插值
'linear'	线性插值
'spline'	三次样条插值
'cubic'	立方插值

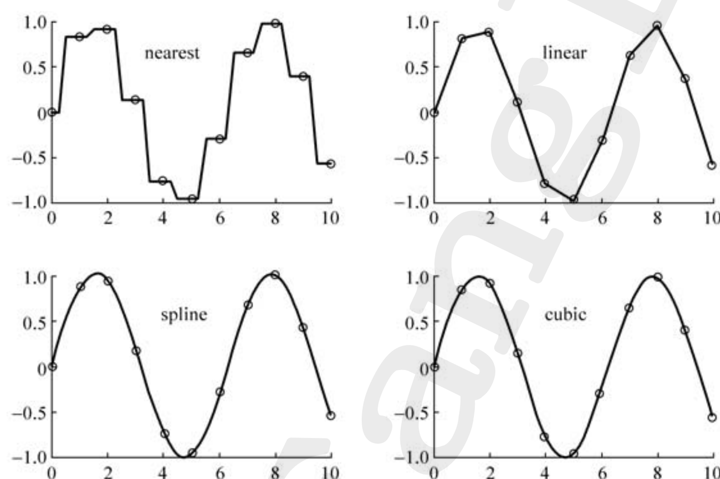


图 1.7 四种插值方法的计算结果

在图 1.7 中，我们用正弦函数的 10 个值作为样点数据组。再分别用四种方法插入 41 个函数插值并画出曲线，原始数组则以小圆圈表示。实际程序 chazhi.m 如下：

```
x = 0:10; y = sin(x); xi = 0:.25:10
yi1 = interp1(x,y,xi,'nearest');
yi2 = interp1(x,y,xi,'linear');
yi3 = interp1(x,y,xi,'spline');
yi4 = interp1(x,y,xi,'cubic');
subplot(2,2,1)
plot(xi,yi1,x,y,'o')
subplot(2,2,2)
plot(xi,yi2,x,y,'o')
subplot(2,2,3)
plot(xi,yi3,x,y,'o')
```

```
subplot(2,2,4)
plot(xi,yi4,x,y,'o')
```

为了更好地对比，在图 1.8 中将它们合并在一幅图上。

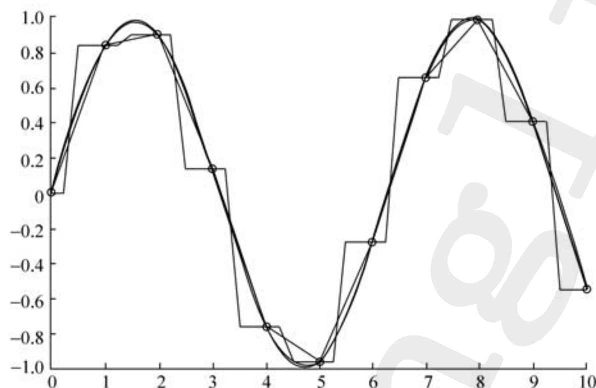


图 1.8 四种插值方法的对比

(2) interp1q 快速一维线性插值

调用格式为

yi=interp1q(x, y, xi)

各项符号的含义与上面相同。它采用的是线性插值算法。要求 x 是单调增加的矢量， y 是相同长度的矢量或是行数与 x 长度相同的矩阵。它适用于不等间距的数据，因为它不进行输入校验，所以计算速度比 interp1(···, 'linear') 更快。如果数据是等间距的，则是 interp1(···, 'linear') 的计算速度更快。

(3) interp2 二维插值

调用格式有四种：

zi=interp2(x, y, z, xi, yi)

zi=interp2(z, xi, yi)

zi=interp2(z, ntimes)

zi=interp2(··· , 'method')

各项符号的含义与上面相似。 ntimes 表示用迭代法进行 n 次内插计算。

三维插值指令 interp3 和 N 维插值指令 interpn 的用法与此类似，不再重复。

除了上面介绍的几条指令， spline 也是插值计算的指令，还有插值工具箱 (spline toolbox) 可资利用，在工具箱中有许多演示例子，可以用指令 splexmpl, csapsdem, csapidem, chebdem, spcrvdem 调用。

2. polyfit 曲线的多项式拟合

对数据作最小二乘法的多项式拟合的指令是 polyfit, 语句格式是

polyfit (X, Y, N)

其中 X, Y 是样点数据， N 是拟合多项式的最高幂次，返回的是拟合多项式的系数，按降幂排列。这个指令可用别的输出格式并配合指令 polyval 来估计拟合的误差，要深入了解请查 help 。

下面是一个简单的程序 nh.m ，首先用二次多项式 $0.4t^2+2t$ 建造了一组数据 s ，然后给它加上一些偏差值 s1 得到数据组 ss, 最后对它作多项式拟合并将结果画成曲线即图 1.9 。图中数据组 ss 用圆圈标出。

```
t=0:0.5:5;
s=2*t+0.4.*t.*t;
s1=[0.05 -0.03 -0.01 0.03 0.01 ...
    0.01 -0.02 -0.01 0.02 0.03 0.01];
ss=s+s1;
P=polyfit(t,ss,2);
y=polyval(P,t);
plot(t,y,t,ss,'o')
```

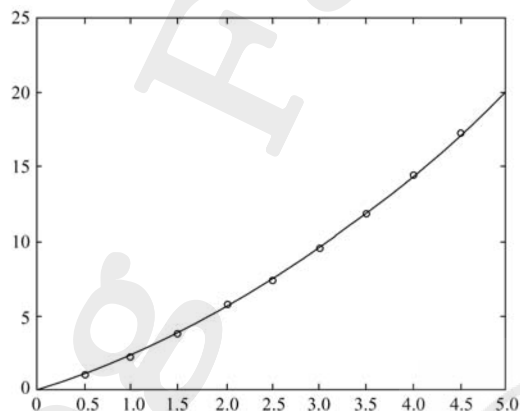


图 1.9 用多项式拟合的曲线

1.4.2 快速傅里叶变换

快速傅里叶变换 (FFT) 是离散傅里叶变换 (DFT) 的一种快速算法。MATLAB 使用的变换公式是

$$X(k)=\sum_{n=1}^{N}x(n)\mathrm{e}^{j2\,\pi\,(k-1)(\frac{n-1}{N})}\qquad(1\le k\le N)$$

$$x(n)=\frac{1}{N}\sum_{k=1}^{N}X(k)\mathrm{e}^{-j2\,\pi\,(k-1)(\frac{n-1}{N})}\qquad(1\le n\le N)$$

长度为 N 的数据系列 $\mathbf{x}$ 的变换的结果是长度为 N 的矢量 $\mathbf{X}$ 。此处公式中求和指标是从 1 到 N, 而常见的公式中求和是从 0 到 $N-1$, 这是因为矩阵指标不能为零。

如果 $x(n)$ 为实数，可以将公式改写为实系数的形式：

$$x(n)=\frac{1}{N}\sum_{k=1}^{N}a(k)\cos\left(\frac{2\,\pi\,(k-1)(n-1)}{N}\right)+b(k)\sin\left(\frac{2\,\pi\,(k-1)(n-1)}{N}\right)$$

其中

$$a(k)=\mathrm{Re}(X(k)),\quad b(k)=-\mathrm{Im}(X(k))\qquad(1\le n\le N)$$

进行 FFT 的指令有一维的 FFT 指令及其逆变换指令 fft, ifft, 二维的 FFT 指令及其逆变换指令 fft2, ifft2, n 维的 FFT 指令及其逆变换指令 fftn, ifftn 。

1. fft, ifft　一维的 FFT 及其逆变换

其语句格式为

fft(x)

fft(x, N)

fft(x, [ ], dim)

fft(x, N, dim)

在这里， x 是待变换的数据，如果输入数据的长度是 2 的整次幂，则 FFT 自动采用以 2 为基数的快速算法，否则就采用较慢的算法。如果 x 是矩阵， FFT 是对每一列进行；如果 x 是多维列阵， FFT 只对在前面的维的元素数不是 1 的维进行。 N 表示进行 N 个点的 FFT ，如果 N 大于 x 的长度，则用零补充；如果 x 大于 N 的长度，则把多余的部分去掉。后两条格式的功能是相同的， FFT 只对 dim 指定的维进行。逆变换 ifft 的格式与 fft 的格式相同。

2. fft2, ifft2　二维的 FFT 及其逆变换

其语句格式为

fft2(x)

fft2(x, mrows, ncols)

在这里， x 是要变换的二维矩阵，如果它是一维的，则按一维进行变换。mrows, ncols 表示在变换之前，通过补零或截断的办法，将 x 的大小改造成 m×n 的矩阵。逆变换 ifft2 的格式与 fft2 的格式相同。

n 维的 FFT 及其逆变换指令 fftn, ifftn 的格式分别与 fft2, ifft2 的格式相同。

下面的程序 fly.m 是一个傅里叶变换的例子。

```
x = [4 3 7 -5 1 -3 -4 2]′;
Y=fft(x)
X=ifft(Y)
Y(1)=[];      %%去掉零频分量
n=fix(length(Y)/2);      %%计算不同频率的数目
freq=(1:n)/length(Y);      %%计算频率
power=abs(Y(1:n)).^2/(length(Y).^2);      %%计算功率谱
power=power′
```

程序运行的结果如下：

```
Y =
   5.0000
  12.1924  - 10.2929i
   2.0000  -  3.0000i
  -6.1924  + 11.7071i
  11.0000
  -6.1924  - 11.7071i
   2.0000  +  3.0000i
  12.1924  + 10.2929i
X =
   4.0000
   3.0000  + 0.0000i
   7.0000
  -5.0000  - 0.0000i
   1.0000
  -3.0000  + 0.0000i
  -4.0000
   2.0000  - 0.0000i
freq =
      0.1429    0.2857    0.4286
power =
```



```
     5.1959    0.2653    3.5796
```

这里，实数数组 x 的 FFT 变换是复数 Y，而 Y 的逆变换 X 就是 x。 Y 的第一个分量是 x 的所有分量之和所得的常量，而 y 的第五个分量是 Nyquist 频率。 y 的后三个分量对应负频，对于实数 x，它们是 y 的前一半中的三个频率分量的复共轭。按照变换公式可以求出各个频率所对应的功率谱。

通过本节的学习，可以看出 MATLAB 有以下特点：同一个任务可以有多条指令；同一条指令可有多种调用格式和多种算法；不同算法有不同的精度。这说明 MATLAB 的功能强大和完善，但也容易使初学者眼花缭乱。事先了解这一点，才不会被名目众多的指令弄得无所适从。

1.4.3 求极大值、极小值和零点

MATLAB 可以对函数求极小值和根，还可以用插值法求矢量和矩阵的极大值，找矩阵的最大元素，求多项式的根。在优化工具箱 (Optimization Toolbox) 中有许多有关的指令。键入 help optim 可以查看。下面介绍几条常用的指令：

1. fminbnd　对单变量函数求极小值

语句格式和各项符号说明如下：

x = fminbnd(fun, x1, x2)

x = fminbnd(fun, x1, x2, options)

x = fminbnd(fun, x1, x2, options, p1, p2, ⋯)

[x, fval] = fminbnd(⋯)

[x, fval, exitflag, output] = fminbnd(⋯)

[x, fval, exitflag, output] = fminbnd(⋯)

x, fval	与极小值对应的 x 值和函数值。
fun	函数名，用 M 文件建立的要加引号，指令 inline 建立的不加。
x1, x2	自变量取值范围。
options	控制算法的优化选项，有四项（见 1.4.5 节）。[ ] 是默认值。
Display	表示计算过程是否显示，'iter' 显示中间的所有步骤，'final' 只显示最后一步，'off' 为不显示。
Tolx	x 的允许误差，默认值是 0.0001。
TolFun	函数值的允许误差。
MaxFunEvals	规定最多计算多少次函数值，默认值是 500。
MaxIter	规定最多进行多少次的插值计算。
p1, p2, ⋯	向函数传递的参数。
exitflag	表示中止搜索的条件。 1 是根据 TolFun 得到的 x 的收敛

	值， 0 是已达到 MaxFunEvals 要求的函数值的计算个数。
output	已执行的迭代计算的次数。

2. fminsearch 对多变量函数求极小值

语句格式和各项符号说明如下：

x = fminsearch(fun, x0)

x = fminsearch(fun, x0, options)

x = fminsearch(fun, x0, options, p1, p2, ⋯)

[x, fval] = fminsearch(⋯)

[x, fval, exitflag] = fminsearch(⋯)

fun	用 M 文件建立的函数。
x0	猜测的初始值。
x , fval	局部极小值对应的自变量的值和函数的值。
p1, p2, ⋯	传递给函数的参量。
options	控制算法的优化选项，有五项。 [ ] 是默认值，后面有介绍。
Display	表示计算过程是否显示。
TolFun	函数值的允许误差。
TolX	x 允许的误差值，默认值是 0.0001 。
TolFun	函数值允许的误差，默认值是 0.0001 。
MaxFunEvals	规定最多计算多少次函数值，默认值是 200*length(x0) 。
MaxIter	规定最多进行多少次的插值计算。
exitflag	停止搜索的条件， 1 是收敛到 x, 0 是达到规定的迭代次数。

3. findmax(z) 对矢量求极大

用三次或二次插值求矢量 z 数据中的极大值。

4. findmax2(z) 对矩阵求极大

用二次插值求矩阵 z 中所有极大值。

5. fzero 对单变量函数求零

语句格式及各项符号说明如下：

x = fzero(fun, x0)

x = fzero(fun, x, options)

x = fzero(fun, x, options, p1, p2, ⋯)

[x, fval] = fzero(⋯)

[x, fval, output] = fzero(⋯)

x	找到的零点，函数值在此会改变符号，没有零点则返回 NaN 。
fval	零点对应的函数值。
fun	用 M 文件或 inline 建立的单变量函数，其函数值必须是实数。
x0	猜测的初始值或搜寻零点的区间 [x0(1), x0(2)]，必须是实数。如是区间，要求函数值在 x0 (1) 与 x0 (2) 符号相反。如是猜测值，则 fzero 将寻找一个包含 x0 同时函数值符号发生改变的区间，如果寻找中遇到了 Inf， NaN 或复数，则寻找被中止。
options	控制算法的优化选项，可选 Display 和 TolX 两项， [ ] 是默认值。
p1, p2	传递给函数的参数。
exitflag	找到零点则输出值 >0, 否则输出值 <0 。
output	已执行的迭代计算的次数。

6. roots(c)　对多项式求零点

指令 roots(c) 对以 c 作为各幂次项的系数所组成多项式求零点。如多项式为

$$c_1x^n + c_2x^{n-1} + \cdots + c_nx + c_{n+1}$$

则 c 的表达式为

$$c = [c_1,\ c_2,\ \cdots,\ c_n,\ c_{n+1}]$$

举例如下：

(1) 建立函数 qx 并画出函数图形（见图 1.10 ）

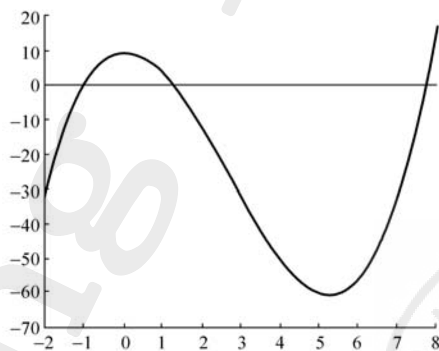


图 1.10　函数 x^3-8x^2+x+10 的图形

```
>> qx = inline('x.^3 - 8*x.^2 + x + 10');
>> fplot(qx, [-2, 8])
```

(2) 求函数的极小

```
>> fminbnd(qx, -2, 8, optimset('Display','off'))
```

ans=

5.2701

(3) 求极大值

先求出对应变量区间 x = [-2 : 0.4 : 8] 的函数值 z, 这是一个矢量。然后用 findmax 求 z 的极大值，它是用插值法求出的，所以不一定是矢量 z 的元素，共得到两个极大值 10.0315 和 18.0000 。为了对比，还用矩阵运算的指令 max 求出了 z 的最大元素 18 。

```
>> z=qx([-2:0.4:8])
z =
    Columns 1 through 7
    -32.0000  -16.1760  -4.4480  3.5680  8.2560  10.0000  9.1840
    Columns 8 through 14
    6.1920  1.4080  -4.7840  -12.0000  -19.8560  -27.9680  -35.9520
    Columns 15 through 20
    -43.4240  -50.0000  -55.2960  -58.9280  -60.5120  -59.6640
    Columns 21 through 26
    -56.0000  -49.1360  -38.6880  -24.2720   -5.5040   18.0000
>> findmax(z)
ans =
     10.0315
     18.0000
>> max(z)
ans =
     18
```

(4) 求函数的零点

对三个不同的零点用三个不同的猜测值，可以从小到大逐步搜索。

```
>> fzero(qx,-0.8)
ans =
       -1
>> fzero(qx,2)
ans =

      1.2984
>> fzero(qx,6)
ans =
```



```
      7.7016
```

如果用符号变量表达式 (见 1.5 节) 就可以把三个根一次求出来，而且是精确解，再用 double 将它转为双精度数。

```
>> GX=sym('x^3-8*x^2+x+10')
GX =
     x^3-8*x^2+x+10
>> GXANS=solve(GX)
GXANS =
        [                -1]
        [ 9/2+1/2*41^(1/2)]
        [ 9/2-1/2*41^(1/2)]
>> double(GXANS)
ans =
      -1.0000
       7.7016
       1.2984
```

(5) 用多项式求根指令找 qx 的根

```
>> roots([1,-8,1,10])
ans =
          7.7016
          1.2984
         -1.0000
```

但是， fzero(inline('abs(x) + 1'), 1) 的结果是 NaN ，因为此时不存在零点。

(6) 用 M 文件来建立函数

下面建立了一个带有参数的函数文件 qx.m 。

```
%% program qx.m
function y = qx(x, a)
y = x.^3 - a*x.^2 + x + 10
```

对文件建立的函数使用以上的命令时, 要给函数名加引号。以求零点为例, 操作如下，各种选项采用了默认值，输出的结果包括找到零点的区间、变量值和函数值。传给函数的参量值为 $a = 8$ 。

```
>> [x, y] = fzero('qx', -0.8, [ ], 8)
Zero found in the interval: [-0.544, -1.056].
x =
    -1
```

```
y =
    0
```

如果用指令 optimset 去改变默认的选项，如要求显示计算过程，则操作如下：

```
>> [x, y] = fzero('qx', -0.8, optimset('disp', 'iter'), 8)
```

(7) 求多元函数极小值

先建立函数文件 threevar.m，

```
function b = threevar(v)
x = v(1);       y = v(2);          z = v(3);
b = x^2 + 2.5*sin(y) - z^2*x^2*y^2;
```

然后在 (-0.6, -1.2, 0.135) 点执行求极小值的指令。

```
>> v = [-0.6  -1.2  0.135];
>> a = fminsearch('threevar',v)
a =
    0.0000    -1.5708    0.1803
```

1.4.4 解方程与方程组

解方程可分为两种，一种是对符号变量方程求解析解；另一种是对非符号变量方程求数值解。

1. solve　求符号变量方程的解析解

符号工具箱中的指令 solve 可以用来求符号变量方程的解析解。下面建立一个符号变量方程，并对它求解。

```
>> syms  a  b  c  x
>> f=a*x^2+b*x+c
>> solve(f)
ans =
      [1/2/a*(-b+(b^2-4*a*c)^(1/2))]
      [1/2/a*(-b-(b^2-4*a*c)^(1/2))]
```

也可以对指定变量求解，如

```
>> b = solve(f,b)
b =
    -(a*x^2+c)/x
```

指令 solve 可以解三角函数方程

```
>> s = solve('cos(2*x)+sin(x)=1')
s =
```



```
    0
    pi
    1/6*pi
    5/6*pi
```

输出的解是有四个分量的一个矢量。再看另一个例子：

```
>> s = solve('x^3 - 2*x^2 = x - 1')
s =
    1/6*(28+84*i*3^(1/2))^(1/3)+14/3/(28+84*i*3^(1/2))^(1/3)+2/3

    -1/12*(28+84*i*3^(1/2))^(1/3)-7/3/(28+84*i*3^(1/2))^(1/3)
        +2/3+1/2*i*3^(1/2)*(1/6*(28+84*i*3^(1/2))^(1/3)
        -14/3/(28+84*i*3^(1/2))^(1/3))

    -1/12*(28+84*i*3^(1/2))^(1/3)-7/3/(28+84*i*3^(1/2))^(1/3)
        +2/3-1/2*i*3^(1/2)*(1/6*(28+84*i*3^(1/2))^(1/3)
        -14/3/(28+84*i*3^(1/2))^(1/3))
```

把它表示成数值形式的指令是 vpa,

```
>> vpa(s, 10)
ans =
      2.246979604   +.1e-9*i
     -0.8019377357  +.3e-9*i
      0.5549581323  -.5e-9*i
```

指令 solve 也能解方程组，设 $r1$, $r2$, $r3$ 为已知量， x, y, z 为未知量，解下面的非线性方程组：

$$\begin{cases} \dfrac{1}{x}+\dfrac{1}{y+z}=\dfrac{1}{r1} \\ \dfrac{1}{y}+\dfrac{1}{z+x}=\dfrac{1}{r2} \\ \dfrac{1}{z}+\dfrac{1}{x+y}=\dfrac{1}{r3} \end{cases}$$

具体的解法如下：

```
>> syms x y z r1 r2 r3
>> [x,y,z]=solve(1/x+1/(y+z)-1/r1,1/y+1/(z+x)-1/r2,1/z+1/(x+y)-1/r3)
x =
    1/2*(-2*r1*r3-2*r2*r1+r1^2-2*r2*r3+r3^2+r2^2)/(r1-r3-r2)
y =
   -1/2*(-2*r1*r3-2*r2*r1+r1^2-2*r2*r3+r3^2+r2^2)/(r1+r3-r2)
z =
```

```
  -1/2*(-2*r1*r3-2*r2*r1+r1^2-2*r2*r3+r3^2+r2^2)/(r1-r3+r2)
```

再解一个方程组：

```
>> syms x y alpha
>> [x,y] = solve(x^2*y^2-1,x-(y/2)-alpha)
x =
    1/2*alpha+1/2*(alpha^2+2)^(1/2)
    1/2*alpha-1/2*(alpha^2+2)^(1/2)
    1/2*alpha+1/2*(alpha^2-2)^(1/2)
    1/2*alpha-1/2*(alpha^2-2)^(1/2)
y =
    -alpha+(alpha^2+2)^(1/2)
    -alpha-(alpha^2+2)^(1/2)
    -alpha+(alpha^2-2)^(1/2)
    -alpha-(alpha^2-2)^(1/2)
```

2. 求线性方程的数值解

如果方程没有解析解或者只要求数值解，那么使用符号工具箱解方程并非最佳选择，而应该使用 MATLAB 本身的数值求解的方法。MATLAB 的做法是将方程组分为线性的和非线性的两类，对形式为 $\boldsymbol{A}\ \boldsymbol{x} = \boldsymbol{b}$ 的线性方程组，（这里 $\boldsymbol{A}$ 是 n × m 的矩阵，$\boldsymbol{b}$ 是已知矢量，$\boldsymbol{x}$ 是未知矢量）可分为三种：

$n = m$ 且 $\boldsymbol{A}$ 非奇异	恰定方程
$n > m$	超定方程
$n < m$	欠定方程

用 MATLAB 求解时都是使用矩阵左除的方法即使用程序式 `x = A\b`。例如要解方程组：

$$\begin{cases} 3x + 5y - 7z = 34 \\ 2x - 12y + 3z = -56 \\ -x + 9y + 8z = 27 \end{cases}$$

相应的操作是

```
>> A = [ 3  5 -7; 2  -12  3; -1 9 8 ];
>> b = [ 34; -56; 27];
>> x = A\b
ans =
      0.5474
```

```
 4.3854
-1.4901
```

3. fsolve 求非线性方程的数值解

下面介绍用数值计算法解非线性方程（组）的指令 fsolve 。指令 fsolve 是用最小二乘法解非线性方程 $f(x)=0$ ，命令格式如下：

x = fsolve(fun, x0)

x = fsolve(fun, x0, options)

x = fsolve(fun, x0, options, p1, p2, ···)

[x, fval] = fsolve(fun, x0, ···)

[x, fval, exitflag] = fsolve(fun, x0, ···)

[x, fval, exitflag, output] = fsolve(fun, x0, ···)

[x, fval, exitflag, output, jacob] = fsolve(fun, x0, ···)

各项符号的意义为

x,fval 零点的位置与对应的函数值， f 和 x 可以是矢量或矩阵。

fun 与非线性方程对应的函数，方程通常用 M 文件建立。

x0 猜测的初始解，作为求解过程的出发点。

options 控制算法的优化选项， [ ] 是默认值。参见后面介绍。

p1, p2 向函数传递的参量。

exitflag 中止求解的条件。大于零是找到收敛解 x ，等于零是已经达到设定的计算函数值的次数，小于零是没有找到收敛解。

output 输出过程信息，如插值计算和求函数值的次数，所用的算法等。

jacob 返回函数在 x 的 jacob 行列式， jacob 行列式是函数梯度的转置。

下面解方程组：

$$\begin{cases} 2\cos(x)+\sqrt{y}-\ln(z)=7 \\ 2^x+2y-8z=-1 \\ x+y-\cosh(z)=0 \end{cases}$$

首先要编写函数文件，将方程中的 x,y,z 看作矢量 $\mathbf{X}$ 的三个分量，编好函数文件 qg.m 。

```
%% program qg.m
function F = qg(X)
F = [2*cos(X(1)) + sqrt(X(2)) - log(X(3)) - 7;
      2^X(1) + 2*X(2) - 8*X(3) + 1;
      X(1) + X(2) - cosh(X(3))]
```

任意猜测一个初始解 x_0:

$$x_0 = [\ 0.5{+}0.5\mathrm{i},\ 0.5{+}0.5\mathrm{i},\ 0.5{+}0.5\mathrm{i}\ ]$$

求解过程如下:

```
    >> [X,Y] = fsolve ('qg', [0.5+0.5i, 0.5+0.5i, 0.5+0.5i])
X =
    -0.0144 + 1.6009i   1.0568 - 1.7463i   0.4442 - 0.3257i
Y =
    -0.0003 - 0.0005i
     0.0001 + 0.0001i
    -0.0001 + 0.0016i
```

求解时，要求输出零点的 $\boldsymbol{X}$ 值和函数值 $\boldsymbol{Y}$。函数值 $\boldsymbol{Y}$ 各个分量代表的含义是

$$\begin{cases} Y(1) = 2\cos(x) + \sqrt{y} - \ln(z) - 7 \\ Y(2) = 2^x + 2y - 8z + 1 \\ Y(3) = x + y - \cosh(z) \end{cases}$$

当 $\boldsymbol{Y}$ 的三个分量都为零时的 $\boldsymbol{X}$ 值，就是方程的解。求解条件（优化选项）的选择使用默认值（见 1.4.5 节），它对函数值的误差要求是 10^{-6}。从 $\boldsymbol{Y}$ 的表达式可以看出，在最小二乘法的意义上，所得的解 $\boldsymbol{X}$ 是符合要求的。应该注意，猜测的初始值如果离零点较远，很可能由于在求解条件的选择中某个条件的限制而不能求出零点，这时可以改变初始值或者改变求解条件的参数，重新求解，直到找到满意的解为止。

1.4.5 指令中的选项

在使用 fzero, fminbnd, fminsearch, fsolve 等指令时，往往需要说明使用的条件，如给定解的精度，指定求解过程的显示方式即求解过程是否显示出来等。在 MATLAB 中，这些都有默认设置，但可以用 option 中的选项来更改原来的默认设置。这是一些很重要的内容，如果不能正确地使用它们，可能得不到预期的结果。

有关选项的指令有 optimset 和 optimget 。

1. optimset　建立和修改结构数组“优化选项”

语句格式如下:

options = optimset('param1', value1, 'param2', value2, ⋯)

options = optimset(oldopts, 'param1', value1, ⋯)

options = optimset(oldopts, newopts)

options = optimset(optimfunction)

options = optimset

optimset

各项符号说明如下：

options	结构数组“优化选项”的名称；
'param1', value1	选项中的参数 1 及其取值；
'param2', value2	选项中的参数 2 及其取值；
oldopts, newopts	旧的优化选项、新的优化选项；
optimfunction	使用优化选项的函数。

上面的各种使用格式中，第一种格式设置优化选项的方法是指定各个参数的取值，对不指定取值的参数，将被设置 [ ]，它表示取默认值。在不引起混淆的情况下，参数名可以只键入前面的几个字母，也不必区分大小写，如用 'tolf ' 表示 'TolFun' 。但数值的输入必须格式正确，否则仍然采用默认值。

第二种格式使用了原来的优化选项，但对其中的参数 1 等指定了新值。

第三种格式合并两个优化选项 oldopts 和 newopts，重复部分取 newopts 的值。

第四种格式是将优化选项的值赋予 options, 如 options = optimset ('fminbnd')。

第五种格式建立了一个选项结构数组，其中的全部参数都设置成 [ ] 。

键入 optimset 可显示如下所示的全部参数的名称及其默认值，其中默认值用 {} 表示：

```
   DerivativeCheck: [ on | {off} ]
       Diagnostics: [ on | {off} ]
     DiffMaxChange: [ positive scalar {1e-1} ]
     DiffMinChange: [ positive scalar {1e-8} ]
           Display: [ off | iter | {final} ]
 GoalsExactAchieve: [ positive scalar | {0} ]
        GradConstr: [ on | {off} ]
           GradObj: [ on | {off} ]
           Hessian: [ on | {off} ]
       HessPattern: [ sparse matrix ]
        HessUpdate: [ dfp | gillmurray | steepdesc | {bfgs} ]
      JacobPattern: [ sparse matrix ]
          Jacobian: [ on | {off} ]
        LargeScale: [ {on} | off ]
```

```
 LevenbergMarquardt: [ on | off ]
     LineSearchType: [ cubicpoly | {quadcubic} ]
        MaxFunEvals: [ positive scalar ]
            MaxIter: [ positive scalar ]
         MaxPCGIter: [ positive scalar ]
      MeritFunction: [ singleobj | multiobj ]
          MinAbsMax: [ positive scalar | {0} ]
  PrecondBandWidth: [ positive scalar | Inf ]
             TolCon: [ positive scalar ]
             TolFun: [ positive scalar ]
             TolPCG: [ positive scalar ]
               TolX: [ positive scalar ]
           TypicalX: [ vector ]
```

键入 help optimset 可查看全部参数的说明, 这里对其中几个参数说明如下：

Display	显示方式，分不显示、过程显示、最终显示。
MaxFunEvals	规定函数值的最多计算次数。
MaxIter	最多迭代次数。
TolFun	最终的函数值的误差。
TolX	最终的变量值的误差。

2. optimget　获取优化选项中参数取值的信息

有两种语句格式：

val = optimget(options, 'name')

val = optimget(options, 'names', default)

在这里， options 是由 optimset 所设定的值。第一种格式是从优化选项中提取由 name 指定的参数的取值。如果该参数没有指定值，则返回空矩阵[ ]。

第二种格式是从优化选项中提取指定的参数的取值。如果该参数没有指定值，则返回默认值。例如

```
>> val = optimget(opts, 'TolX', 1e-4)
```

当 TolX 没有指定值时，得到 val = 1e−4 。

1.4.6 差分、微分、梯度和拉普拉斯算符

本节介绍计算差分 $\Delta x = x_i - x_{i-1}$ ，求导 $\frac{\mathrm{d}y}{\mathrm{d}x}$ ，求梯度 $\frac{\mathrm{d}f}{\mathrm{d}x} + \frac{\mathrm{d}f}{\mathrm{d}y}$ ，求拉普拉斯算子表示的二阶导数 $\Delta f = \frac{\mathrm{d}^2 f}{\mathrm{d}x^2} + \frac{\mathrm{d}^2 f}{\mathrm{d}y^2}$ 等的指令。

1. diff 计算差分

语句格式为：

diff(X)

diff(X, N)

diff(X, N, DIM)

各项符号含义为：

如果 X 是列 (行) 矢量， diff(X) 的计算式是 [X(2)-X(1), X(3)-X(2), ⋯, X(n)-X(n-1)]，即后项减前项；如果 X 是矩阵， diff(X) 是对矩阵的行矢量作差分计算所得到的矩阵 [X(2:n,:) - X(1:n-1,:)]，即后行减前行。

对 N 维列阵，是沿着 X 的维元素数不为 1 的维进行差分计算。

指令 diff(X, N) 是对矩阵 X 的列矢量计算 N 阶差分， N 应小于或等于矩阵列矢量的元素数。

指令 diff(X, N, DIM) 是对矩阵 X 中由 DIM 代表的维作差分计算，如果 N 大于或等于 DIM 的维元素数，则返回空矩阵。

下面列举的例子表明， diff(X, 1, 1) 表示对 X 的每列进行一阶差分计算，即所得结果是后行减前行，这个结果与 diff(X) 和 diff(X, 1) 计算的结果是一样的。 diff(X, 1, 2) 是对 X 的每行进行一阶差分计算即所得结果是后列减前列；diff(X,2,2) 是对 X 的每行进行二阶差分计算；而 diff(X,3,2) 是对每行作三阶差分计算，阶数正好等于行的元素数，结果是得到一个空矩阵。

```
>> X = [ 3  7  5
         0  9  2];
>> diff(X)
ans =
      -3  2  -3
>> diff ( X , 1 , 1 )
ans =
      -3  2  -3
>> diff ( X , 1 , 2 )
ans =
      4  -2
      9  -7
>> diff(X,2,2)
ans =
        -6
      -16
```



```
>> diff(X,3,2)
ans =
      Empty matrix: 2-by-0
```

差分可以用来作导数的数值近似计算，如在下面的程序 cf.m 中， y1 就是用差分计算 $\sin x^2$ 的导数， y2 则是用导数公式来计算 $\sin x^2$ 的导数，二者之差 y 很小，可以从画出的曲线看出来。由于差分的结果比用导数公式计算的结果少一个元素，故画图时少取了一个元素。此外，用后面介绍的计算梯度的指令 gradient 也可以计算导数， y3 就是用它计算的，从图形可以看出， y3 和 y2 的差别也很小。

```
% program  cf
h =0.001;
x = 0 : h : pi;
y1=diff(sin(x.^2))/h;
y2=2*cos(x.^2).*x;
y=y1-y2;
figure
plot(y1-y2(1:end-1))
hold on
y3=gradient(sin(x.^2),h);
plot(y3-y2)
```

2. gradient 计算矩阵或多维列阵的梯度

语句格式为

[fx, fy] = gradient(f)
[fx, fy] = gradient(f, h)
[fx, fy] = gradient(F, hx, hy)
[fx, fy, fz] = gradient(f)
[fx, fy, fz] = gradient(f, hx, hy, hz)
[fx, fy, fz, ⋯] = gradient(f, ⋯)

各项符号含义为

fx	代表 df/dx，是 x（列）方向的偏微分。
fy	代表 df/dy，是 y（行）方向的偏微分。
fz	代表 df/dz，是 z（层）方向的偏微分。
f	求梯度的矩阵或列阵。
h	标量，是各个方向的步长。不指定 h 的值则取默认值 1 。
hx,hy,hz	标量或矢量，分别表示在 x, y, z 方向上的步长。当它们是矢

量时，其长度应与 f 对应的维的元素数相等。

注意，对矢量用 gradient 作梯度计算所得的结果比用 diff 作差分计算所得的结果多一个元素，同样用它们对矩阵作运算时也有这种差别。当 f 是矢量时，df =gradient(f,h) 可以作导数的近似数值计算公式。

例 电量为 $1/4\pi\epsilon_0$ 的正负电荷分别位于平面上的 (0.2, 0) 和 (−0.2, 0) 处，形成一个电偶极子。下面的程序 qtd.m 画出了电偶极子的等势面(见图 1.11)。电偶极子的电势为 $\boldsymbol{Z}$，利用指令 gradient 计算它在两个方向的梯度，指令 contour 是画等高线，在这里画出的是等电位面，指令 quiver 则画出各点的电场的方向和大小。

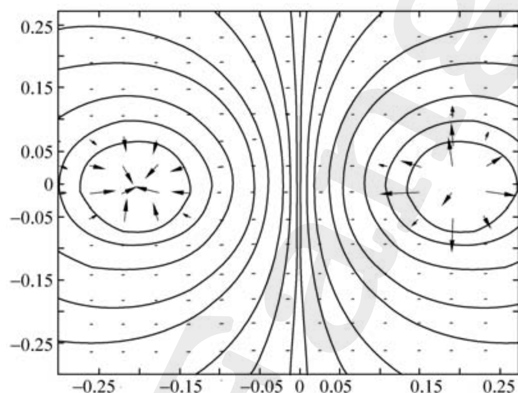


图 1.11 电偶极子的等势面

```
%program  qtd.m
h=0.041; x=-0.3:h:0.3; y=-0.3:h:0.3;
[X,Y]=meshgrid(x,y);
Z=1./sqrt((X-0.2).^2.+Y.^2)-1./sqrt((X+0.2).^2+Y.^2);
[PX,PY]=gradient(-Z,h);
contour(x,y,Z,[-12,-8,-5,-3,-2,-1,-.5,-.1,.1,.5,1,2,3,5,8,12],'b')
hold on
quiver(X,Y,PX,PY,'k')
```

3. del2(U) 离散拉普拉斯算符

把矩阵 $\boldsymbol{U}$ 看成在矩形网格上每点赋值的函数 $u(x,y)$， del2(U) 的定义为

$$L = 0.25\Delta_2 u = \frac{1}{4}\left(\frac{\mathrm{d}^2u}{\mathrm{d}x^2} + \frac{\mathrm{d}^2u}{\mathrm{d}y^2}\right)$$

$\boldsymbol{L}$ 是一个与 $\boldsymbol{U}$ 大小相同的矩阵，矩阵内部的各个元素是

$$L_{i,j} = \frac{1}{4}(u_{i+1,j} + u_{i-1,j} + u_{i,j+1} + u_{i,j-1}) - u_{i,j}$$

在矩形网格的边缘则使用三次方的外推法来使用这个公式。对有 N 个变量的多元函数 u(x, y, z, ⋯), del2(U) 表示

$$L = \frac{1}{2N}\left(\frac{\mathrm{d}^2 u}{\mathrm{d}x^2} + \frac{\mathrm{d}^2 u}{\mathrm{d}y^2} + \frac{\mathrm{d}^2 u}{\mathrm{d}z^2} + \cdots\right)$$

其语句格式为

L = del2(U)

L = del2(U, h)

L = del2(U, hx, hy)

L = del2(U, hx, hy, hz, ⋯)

各项符号意义为

U	N 维列阵。
h	是步长，不指定 h 的值则取默认值 1 。
hx, hy, hz	分别是 x, y, z 相应的步长。标量表示步长相同，矢量则表示每步步长， hx, hy, hz 要与 U 对应的元素个数相等。

下面的例子表明，对函数 $U(x,y) = (x^2+y^2)$ 有 $\Delta_2 U = 4$，而程序式 4∗del2(U) 算得的结果也是 4，可见两者是等效的。

```
[x,y] = meshgrid( -2:2, -3:3 );
U = x.*x + y.*y
U =
    13    10     9    10    13
     8     5     4     5     8
     5     2     1     2     5
     4     1     0     1     4
     5     2     1     2     5
     8     5     4     5     8
    13    10     9    10    13
V = 4*del2(U)
V =
    4      4      4      4      4
    4      4      4      4      4
    4      4      4      4      4
    4      4      4      4      4
```



```
     4     4     4     4     4
     4     4     4     4     4
     4     4     4     4     4
```

1.4.7 积分

利用 MATLAB 的指令可对矢量、矩阵和多维列阵进行梯形积分；对函数进行自适应法的辛普生积分和柯特积分，还可以作二重积分。

1. trapz　*梯形积分*

语句格式为

z=trapz(y)

z=trapz(x, y)

z=trapz(x, y, dim)

其含义为，trapz(y) 计算 y 的数值梯形积分，步长默认为 1，如果步长不是 1 而是 h，则要将 z 乘以步长的值 h。如果 y 是矢量，trapz(y) 是一个数；如果 y 是矩阵，trapz(y) 对 y 的每个列矢量作积分，结果是个行矢量；如果 y 是一个 N 维列阵，计算从维的元素数不是 1 的维开始。

trapz(x, y) 计算 y 对 x 的梯形积分，x 和 y 的长度必须相等。

trapz(x, y, dim) 对 dim 指定的 y 的维进行积分，x 的长度必须是 size(y, dim)。

例如：要对函数

$$y = x^2 - 1 \qquad (x = 0:0.1:1.5)$$

作梯形积分。操作如下：

```
>> x = 0:0.1:1.5;
>> y = x.^2-1;
>> I = 0.1*trapz(y)
I =
    -0.3725
```

又如

```
>> Y = [ 0  1  2
         3  4  5 ];
>> trapz(Y,1)
ans =
         1.5000     2.5000     3.5000
>> trapz(Y, 2)
```

```
ans =
      2
      8
```

2. cumtrapz　累计梯形积分

语句格式为

z = cumtrapz(y)

z = cumtrapz(x, y)

z = cumtrapz(x, y, dim)

如果 y 是矢量， z = cumtrapz(y) 也是一个矢量。 z 的第 n 个元素是 y 的前 n 个元素的梯形积分。因此，它的作用很像是 y 的原函数。类似地，如果 y 是矩阵， cumtrapz(y) 也是一个矩阵。如果 y 是一个 N 维列阵，计算从维元素数不是 1 的维开始。

用 cumtrapz(y) 计算累计梯形积分时，步长默认为 1 ，如果步长不是 1 ，则要将 z 乘以步长的值。

用 cumtrapz(x, y) 计算 y 对 x 的累计梯形积分， x 和 y 的长度必须匹配。

用 cumtrapz(x, y, dim) 对 dim 指定的 y 的维进行累计梯形积分， x 的长度必须是 size(y, dim) 。

下面的例子是做累积积分：

```
>> Y = [-1     5     2
         3     6     7
        -7     5    -3];
>> cumtrapz(Y, 1)
ans =
         0         0         0
    1.0000    5.5000    4.5000
   -1.0000   11.0000    6.5000
>> cumtrapz(Y, 2)
ans =
      0    2.0000    5.5000
      0    4.5000   11.0000
      0   -1.0000         0
```

矩形积分可用指令 sum 或 cumsum 来实现，其结果类似于 trapz 或 cumtrapz ，但精度更低，所以一般不用。

3. quad, quad8　函数积分

quad 和 quad8 都可以对用 M 文件或指令 inline 建立的函数作数值积分，二者的使用语句格式相同，但使用的积分计算方法不同， quad 是用自适应辛普生法，属低阶方法， quad8 用自适应柯特法，属高阶方法，精度较高。

quad 的语句格式为

q = quad('fun', a, b)

q = quad('fun', a, b, tol)

q = quad('fun', a, b, tol, trace)

q = quad('fun', a, b, tol, trace, p1, p2, ⋯)

各项符号含义为

fun	积分函数；
a, b	积分区间；
tol = [rel_ tol abs_tol]	积分的相对精度和绝对精度，默认值为 0.0001 ；
trace	取 0 表示不用图形显示积分过程，非 0 表示用图形显示积分过程；
p1, p2, ⋯	是向函数传递的参数。

为了使用默认值的 tol 或 trace ，可以传递一个空矩阵 [ ] 。当 q=Inf 时，表示积分出现奇异值。将上面的指令由 quad 改成 quad8 ，所有的语句也成立。举例如下：

```
>> qx = inline('x.^3 - a*x.^2 + x + 10','x', 'a')
qx =
     Inline function:
     qx(x,a) = x.^3 - a*x.^2 + x + 10
>> quad8(qx, -2, 7, [ ], [ ], 8)
ans =
      -227.2500
```

4. dblquad 数值二重积分

语句格式为：

result = dblquad('fun', inmin, inmax, outmin, outmax)

result = dblquad('fun', inmin, inmax, outmin, outmax, tol)

result = dblquad('fun', inmin, inmax, outmin, outmax, tol, method)

各项符号的含义为：

fun(inner, outer)	被积分的函数， inner 是内变量， outer 是外变量；
inmin, inmax	内变量的积分区间；
outmin, outmax	外变量的积分区间；
tol	积分精度；

method　　　积分方法，有 quad 或 quad8 两种，默认为 quad 。

下面是一个二元函数的积分：

```
>> f2 =inline('y*sin(x) + x*cos(y)', 'x', 'y')
f2 =
     Inline function:
     f2(x,y) = y*sin(x) + x*cos(y)
>> dblquad(f2, pi, 2*pi, 0, pi)
ans =
      -9.8698
```

1.4.8 解常微分方程组

MATLAB 解常微分方程组的能力很强而且很方便，只要用简单的指令就可以解常微分方程，针对不同类型的常微分方程 MATLAB 有不同的指令。下面先看一个简单的例子。匀加速运动的微分方程是

$$\frac{\mathrm{d}^2x}{\mathrm{d}t^2} = a$$

其中 $a = 4$。MATLAB 要求先将方程写成一阶的常微分方程组，令 $y(1) = x$, $y(2) = \mathrm{d}x/\mathrm{d}t$, 得

$$\begin{cases} \dfrac{\mathrm{d}y(1)}{\mathrm{d}t} = y(2) \\ \dfrac{\mathrm{d}y(2)}{\mathrm{d}t} = 4 \end{cases}$$

将它写成如下形式的函数文件 yjs.m 并存盘，

```
function ydot=yjs(t,y)
ydot=[y(2);
      4   ];
```

在函数文件中要求两个等号左边的变量名相同，这里是用的 ydot ，对比原方程知， ydot(1) 就是 dy(1)/dt ， ydot(2) 就是 dy(2)/dt 。等号右边的名称是存盘文件的名称，这里用的是 yjs ，第一行括号中的 t 是时间变量， y 则是在一阶常微分方程组中定义的量。这些格式不能随便改变。在指令窗口键入

```
>> [T,Y] = ode23('yjs', [0:0.1:10], [2,1]);
>> size(T)
ans =
      101   1
>> size(Y)
ans =
```

```
      101    2
  >> plot(T, Y(:,1), T, Y(:,2))
```

在这里， ode23 是解常微分方程组的指令，在调用时，要说明函数文件名即 yjs, 时间范围即 0 : 0.1 : 10 ，初始位移与初始速度即 2 和 1 。输出的结果是时间变量的值即 T 、位移 Y(:, 1) 与速度 Y(:, 2) 的值。指令 size 是查看 T 和 Y 的大小。最后用指令 plot 画曲线表示方程的解。

以上求解过程主要分为三步，即编写函数文件 (odefile), 选择恰当的解方程的指令，指定解方程的要求并按规定的格式调用它们解方程组。下面逐条介绍。

1. 编写 odefile

首先将求解的常微分方程组编写成特殊格式的函数文件，在 MATLAB 中称之为 odefile 。注意， odefile 的格式不同于一般的函数文件，不可随意改变，甚至变量的顺序和位置都不能任意改变。编写 odefile 的方法可查看命令 odefile 。高阶的常微分方程要先化成一阶的常微分方程组再求解。

通常求解的方程组的形式为 $\mathrm{d}\boldsymbol{y}/\mathrm{d}t = \boldsymbol{F}(t, \boldsymbol{y})$ ，这里 t 是自变量， $\boldsymbol{y}$ 是由待求的 t 的函数构成的矢量。所以最简单的 odefile 的文件格式是

function   ydot = odefile(t,y)

ydot = [在括号内插入 t 和 / 或 y 的函数] ;

在这里， t 是标量， y 是列矢量， ydot 是列矢量 dy/dt 。上面的例子中使用的 odefile 就是这种形式。这种形式的 odefile 十分简单，使用也最多，但它的功能有限。为了在解常微分方程的过程中完成一些特殊的任务， MATLAB 5.3 准备了一个 odefile 的模板，键入

```
>> help odefile
```

在屏幕上会显示如下模板：

```
function varargout = odefile(t,y,flag,p1,p2)
switch flag
case ''                        % Return dy/dt = f(t,y).
  varargout{1} = f(t,y,p1,p2);
case 'init'                    % Return default [tspan,y0,options].
  [varargout{1:3}] = init(p1,p2);
case 'jacobian'                % Return Jacobian matrix df/dy.
  varargout{1} = jacobian(t,y,p1,p2);
case 'jpattern'                % Return sparsity pattern matrix S.
  varargout{1} = jpattern(t,y,p1,p2);
case 'mass'                    % Return mass matrix.
  varargout{1} = mass(t,y,p1,p2);
```

```
  case 'events'                  % Return [value,isterminal,direction].
    [varargout{1:3}] = events(t,y,p1,p2);
  otherwise
    error(['Unknown flag ''' flag '''.']);
  end

% ----------------------------------------------------------------------

  function dydt = f(t,y,p1,p2)
  dydt = < Insert a function of t and/or y, p1, and p2 here. >

% ----------------------------------------------------------------------

  function [tspan,y0,options] = init(p1,p2)
  tspan = < Insert tspan here. >;
  y0 = < Insert y0 here. >;
  options = < Insert options = odeset(...) or [~] here. >;

% ----------------------------------------------------------------------

  function dfdy = jacobian(t,y,p1,p2)
  dfdy = < Insert Jacobian matrix here. >;

% ----------------------------------------------------------------------

  function S = jpattern(t,y,p1,p2)
  S = < Insert Jacobian matrix sparsity pattern here. >;

% ----------------------------------------------------------------------

  function M = mass(t,y,p1,p2)
  M = < Insert mass matrix here. >;

% ----------------------------------------------------------------------
```

```
function [value,isterminal,direction] = events(t,y,p1,p2)
value = < Insert event function vector here. >
isterminal = < Insert logical ISTERMINAL vector here.>;
direction = < Insert DIRECTION vector here.>;
```

在模板中， varargout 是 MATLAB 内部定义的关于输出变量的指令，不能换用其它名称。MATLAB 内部定义的关于输入变量的指令是 nargin，这个指令在例 1 的程序中会用到。 t, y 是方程中的变量和函数，在模板中的位置与顺序不能改变或取消。 p1, p2, ⋯ 则是传递给方程组的附加参量。模板中用 switch 分支语句来控制不同的运行模板的方式，每种方式的执行内容都是用不同的子函数来规定，子函数中< >里的内容要由用户编写。各种方式之间的切换是用变量 flag 来执行。变量 flag 是调用模板时输入的由小写字母组成的字符串，其位置不能改变， flag 的可选项包括：

FLAGS	返回值；
′ ′ (empty)	函数 F(t, y)；
′init′	默认的初始条件和优化选项，如 TSPAN, Y0 和 OPTIONS；
′jacobian′	返回 Jacobian 矩阵 J(t, y) = dF/dy；
′jpattern′	返回稀疏 Jacobian 矩阵；
′mass′	返回解方程组 M∗y′=F(t, y) 的质量矩阵 M, M(t) 或 M(t, y)；
′events′	返回过零点位置的事件的信息。

并非每一个 odefile 都要包括模板中的全部内容，如 ′jacobian′ 只用于给解析的雅可比行列式赋值，而 ′jpattern′ 则是产生数值化的雅可比行列式。在 MATLAB 中有 orbitode, rigidode, vdpode 等一些 odefile 的实例，可以键入

```
<< type  < 文件名 >
```

来查看它们。在本节随后的例子中，也有部分使用 odefile 模板的内容，读者可以参看。灵活地使用模板会在解方程的过程中带来许多方便。

在模板中的选项 ′mass′ 是用于求解 M(t, y)∗y′ = F(t, y) 类型的问题，其中 ′M′, ′M(t)′ 和 ′M(t, y)′ 分别表示常数矩阵、与 t 相关的矩阵和与 t, y 相关的矩阵。默认值是 ′none′。指令 ode23 可以解 M 是非奇异矩阵的问题， ode15s, ode23t 可以解 M 是奇异矩阵的问题。在 MATLAB 中， fem1ode, fem2ode, batonode 是几个这类问题的实例，读者可以用 type 查看它们或用 ode23, ode15s, ode23t 去求解它们，并对比一下结果。

在 MATLAB6.0 中，提供了一些新方法如函数句柄等来完成这些任务。考虑到这些新方法对初学者可能更抽象一些，而且 MATLAB6.0 也兼容了 MATLAB5.3 的这些功能，所以本书仍以模板形式来编程。在 MATLAB6.0 中查看模板 odefile 的方法是在指令窗口中键入

>> more on, odefile, more off

待有关内容显示在屏幕上之后，不断按 Enter 键，就会在屏幕逐页显示 odefile 的全部内容。

2. 确定 odeset 中的选项

为了能够解出方程，要用指令 odeset 确定求解的条件和要求。在 MATLAB 中，求解方程组的指令都有默认的求解的条件和要求，通常以结构数组“优化选项” (options structure) 表示，但可以用 odeset 修改它或重新建立，也可以用 odeget 去获取已有的“优化选项”的信息。指令 odeset 和 odeget 用法介绍如下：

(1) odeset 建立和修改“优化选项”

语句格式如下：

options = odeset('name1', value1, 'name2', value2, ⋯)

options = odeset(oldopts, 'name1', value1, ⋯)

options = odeset(oldopts, newopts)

odeset

各项符号的含义为

options	结构数组“优化选项”的名称；
'name1',value1 ，'name2',value2	选项中的参数 1,2 及其取值；
oldopts,newopts	旧的优化选项，新的优化选项；
optimfunction	使用优化选项的函数。

第一种格式是指定各个参数的取值，对不指定取值的参数，取默认值。在不引起混淆的情况下，参数名可以只键入前面的几个字母，也不必区分大小写，如用 'abst' 表示 'AbsTol'。但数值的输入必须格式正确，否则仍然采用默认值。

第二种格式使用了原来的优化选项，但对其中的参数 1 等指定了新值。

第三种格式合并了两个优化选项 oldopts 和 newopts ，重复部分取 newopts 的指定值。

键入 odeset 可在屏幕上显示如下全部可设置的参数及其默认值（{ } 表示默认值）：

```
      AbsTol: [ positive scalar or vector {1e-6} ]
         BDF: [ on | {off} ]
      Events: [ on | {off} ]
 InitialStep: [ positive scalar ]
    Jacobian: [ on | {off} ]
   JConstant: [ on | {off} ]
    JPattern: [ on | {off} ]
        Mass: [ {none} | M | M(t) | M(t,y) ]
```

```
    MassSingular: [ yes | no | {maybe} ]
        MaxOrder: [ 1 | 2 | 3 | 4 | {5} ]
         MaxStep: [ positive scalar ]
     NormControl: [ on | {off} ]
       OutputFcn: [ string ]
       OutputSel: [ vector of integers ]
          Refine: [ positive integer ]
          RelTol: [ positive scalar {1e-3} ]
           Stats: [ on | {off} ]
      Vectorized: [ on | {off} ]
```

键入 help odeset 可查看全部参数的说明，下面对其中几个参数举例说明。

RelTol　　相对误差，默认值为 1e-3 。

AbsTol　　绝对误差，默认值为 1e-6 。

OutputFcn　　输出方式，默认值为 'odeplot'，其它可选项有：

- odeplot　　按时间顺序画出全部变量的解；
- odephas2　　二维相空间中两个变量的图形；
- odephas3　　三维相空间中三个变量的图形；
- odeprint　　打印输出。

(2) odeget　获取优化选项中参数取值的信息

这个指令有两种语句格式：

val = odeget(options, 'name')

val = odeget(options, 'name', default)

这里 options 是由 odeset 设定的优化选项。第一种格式从优化选项中提取指定的参数的取值。如果该参数没有指定值，则返回空矩阵 [ ] 。

第二种格式从优化选项中提取指定的参数的取值。如果该参数没有指定值，则返回默认值。例如

```
val = odeget(opts, 'RelTol', 1e-4);
```

当 RelTol 没有指定值时，得到 val = 1e−4 。

3. 选择和调用解常微分方程组的指令

解常微分方程组的指令有好几种，应该根据方程的类型来选用。常微分方程分“刚性的”和“非刚性的”，刚性的是指其 Jacobian 矩阵的特征值相差十分悬殊。两者对解法中步长选择的要求不同。下面的解法中，有的可用于解刚性方程，有的则不适用。如果不能分辨是否是刚性方程，先试用 ode45 ，再用 ode15s 。解常微分方程组的指令有：

ode45　解非刚性微分方程，中等精度，使用 Runge-Kutta 法的四、五阶算法。

ode23 解非刚性微分方程，低精度，使用 Runge-Kutta 法的二、三阶算法。
ode113 解非刚性微分方程，变精度变阶次 Adams-Bashforth-Moulton PECE 算法。
ode23t 解中等的刚性微分方程，使用自由内插法的梯形法则。
ode15s 解刚性微分方程，使用可变阶次的数值微分 (NDFs) 算法。
ode23s 解刚性微分方程，低阶方法，使用修正的 Rosenbrock 公式。
ode23tb 解刚性微分方程，低阶方法，使用 TR-BDF2 方法，即 Runger-Kutta 公式的第一级采用梯形法则，第二级采用 Gear 法。

指令的语句格式为 (用 ode23 为例，其余指令用法相似)：

[T, Y] = ode23('F', tspan, y0)

[T, Y] = ode23('F', tspan, y0, options)

[T, Y] = ode23('F', tspan, y0, options, p1, p2, ⋯)

[T, Y, TE, YE, IE] = ode23('F', tspan, y0, options)

其含义为

F 求解的常微分方程的文件名，方程的形式为 y′ = F(t, y) 。
tspan 单调递增（减）的积分区间 [t0 tfinal] 或 [t0, t1, ⋯， tfinal] 。
y0 初始条件矢量。
options 用 odeset 建立的优化选项，如用默认值则不必输入。
p1, p2, ⋯ 传递给 F 的参数，这时用 [ ] 表示默认的优化选项。
T, Y T 是输出的时间列矢量、矩阵 Y 每个列矢量是解的一个分量。
TE, YE, IE 当 y 的某个分量为零时的 t 和 y 的值以及它们的序号。使用此项时要将优化选项中 'events' 设为 'on' ，参见本节的例 2 。

例 1 求解洛仑兹方程。这是混沌理论中一个著名的方程，是一个三变量的耦合的一阶常微分方程组。

$$\begin{cases} \dfrac{\mathrm{d}x}{\mathrm{d}t} = -\dfrac{8}{3}x + yz \\ \dfrac{\mathrm{d}y}{\mathrm{d}t} = -10y + 10z \\ \dfrac{\mathrm{d}z}{\mathrm{d}t} = -xy + 35y - z \end{cases}$$

在这个例子中函数文件是 lorfun.m ，解方程的程序文件是 lor.m ，都列出在此段落后面。这是两个不同的文件，需要分开输入，初学者往往会把它误认为是一个文件而产生错误。在调用 ode23 时，除了要求给出 t 的区间和三个函数的初值之外，还要求设定输出三维空间的相图，输出方式是用 odeset 来设置的。此外，对坐标轴的区间、视角和标题也作了规定。画出的图用圆圈标出了每个数据点的位置，用图形窗口的操作可以把圆圈去掉（见 1.5.1 节），就得到图 1.12 。如果采用默认的输出设置，可以将调用 ode23 的语句改为

```
ode23('lorfun',[0,20],[0,0,eps])
```

如果要输出 t, y 的数据，可以将调用 ode23 的语句改为

```
[t,y]=ode23('lorfun',[0,20],[0,0,eps]
```

读者可以比较一下这几种不同的输出结果。

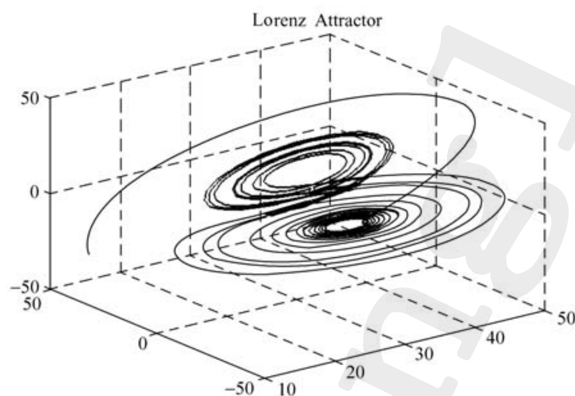


图 1.12 洛仑兹方程的解

```
%%program lor.m
axis([10 50 -50 50 -50 50])
view(3)
hold on
title('Lorenz Attractor')
options =odeset('OutputFcn','odephas3') ;
[t,y]=ode23('lorfun',[0,20],[0,0,eps],options);

%%odefile  lorfun.m
function ydot=lorfun(t,y)
ydot=[-8/3*y(1)+y(2)*y(3);
     -10*y(2)+10*y(3);
     -y(2)*y(1)+35*y(2)-y(3)];
```

利用 odefile 模板可以将上面两个程序合并成一个程序 lor1.m 如下：

```
function varargout = lor1(t,y,flag)
if nargin == 0,
   flag = 'demo';
end
```

```
switch flag
case ' '                          % Return dy/dt = f(t,y).
   varargout{1} = f(t,y);
case 'demo'                       % Run a demo.
   demo;
otherwise
   error(['Unknown flag ''' flag '''.']);
end

function dydt = f(t,y)
A = [ -8/3     0     y(2)
         0   -10      10
      -y(2)   35      -1  ];
dydt = A*y;

function demo
axis([10 50 -50 50 -50 50]);
view(3)
hold on
title('Lorenz Attractor')

options =odeset( 'OutputFcn', 'odephas3' ) ;
[t,y]=ode23( 'lor1', [0,20], [0,0,eps], options);
```

这是一个自身递归调用的程序，涉及到分支语句的使用。其运算过程是：键入 lor1，由于输入变量为零即 nargin 为零，所以执行 flag = 'demo'。在 switch 语句的控制下，再执行子函数 demo, 它设定了坐标轴的取值范围和视角，还有解微分方程的优化选项。最后调用解微分方程的指令 ode23。调用中给定了各种必要的数据，而执行 ode23 又要重新调用程序 lor1, 此时 t 、 y 的初值已经给定，只是 flag 的值为空，所以在 switch 语句的控制下，执行的是子函数 f，即输出待解微分方程，这时 lor1 的作用已经相当于本节开头所介绍的那种简单的函数文件。这个过程不断循环，直到方程被解出，最后按照 options 的设定将结果输出。

在 MATLAB6.0 中，将两个程序合并的方法更简单。只须编写如下形式的程序 loren 即可：

```
function loren.m
axis([10 50 -50 50 -50 50])
view(3)
```

```
hold on
title('Lorenz Attractor')
options =odeset('OutputFcn','odephas3') ;
[t,y]=ode23(@lfun,[0,20],[0,0,eps],options);
function ydot=lfun(t,y)
ydot=[-8/3*y(1)+y(2)*y(3);
     -10*y(2)+10*y(3);
     -y(2)*y(1)+35*y(2)-y(3)];
```

在这个程序中，使用了 MATLAB6.0 提供的一种新的数据类型——函数句柄。在程序中使用 ode23 解常微分方程时，调用描述微分方程的函数 lfun 不是用函数名 lfun，而是用函数句柄 @lfun 。同样，本书第二章的程序也都可以仿照这个例子改成用函数句柄调用而不用函数名调用，从而将两个程序合并为一个程序。但这种方法只能在 MATLAB6.0 中使用。由于 MATLAB5.3 仍在广泛使用，为了方便这些读者使用，我们还是保留函数名调用方式而将程序分为两个程序。

例 2 研究有空气阻力时抛体运动的特征。比较下面三种情况下的抛体的轨道：没有空气阻力；空气阻力与速度一次方成正比；以及空气阻力与速度二次方成正比。本例介绍了如何将高阶的微分方程写成一阶微分方程组和如何在解微分方程的指令中传递参数。

以地面为参考系，以抛出点为原点 O 建立直角坐标系 OXY， OX 沿水平方向，OY 竖直向上。质点受重力和空气阻力作用，而空气阻力包括三种情况。质点的运动微分方程可统一表示为

$$m\frac{\mathrm{d}^2\boldsymbol{r}}{\mathrm{d}t^2}=-m\mathbf{g}-b|v|^{p/2}\mathbf{v}$$

空气阻力的三种情况分别对应方程中参数值为

$$b\ =\ [0,\ 0.1,\ 0.1],\ \ p\ =\ [0,\ 0,\ 1]$$

令 $y(1)=x,\ y(2)=\mathrm{d}x/\mathrm{d}t,\ y(3)=y,\ y(4)=\mathrm{d}y/\mathrm{d}t$, 将方程写成一阶微分方程组的形式就是

$$\begin{cases}\dfrac{\mathrm{d}y(1)}{\mathrm{d}t}=y(2)\\ \dfrac{\mathrm{d}y(2)}{\mathrm{d}t}=-\dfrac{by(2)}{m}(y^2(2)+y^2(4))^{p/2}\\ \dfrac{\mathrm{d}y(3)}{\mathrm{d}t}=y(4)\\ \dfrac{\mathrm{d}y(4)}{\mathrm{d}t}=-g-\dfrac{by(4)}{m}(y^2(2)+y^2(4))^{p/2}\end{cases}$$

再用指令 ode45 解常微分方程。主程序 ddqxn.m 如下：

```
m=1;b=[0,0.1,0.1];p=[0,0,1];        %%设置参数
figure
axis([0 9 0 4])      %%设置坐标轴的范围
hold on
for i=1:3      %%解微分方程三次
[t,y]=ode45('ddqxnfun',[0:0.001:2],[0,5,0,8],[ ],b(i),p(i),m);
comet(y(:,1),y(:,3))      %%画轨道的彗星轨迹
end
```

函数文件 ddqxnfun.m 如下：

```
function ydot=ddqxnfun(t,y,flag,b,p,m)
ydot=[y(2);
     -b/m*y(2)*(y(2).^2+y(4).^2)^(p/2);
     y(4);
     -9.8-b/m*y(4)*(y(2).^2+y(4).^2)^(p/2)];
```

例 3 小球的弹跳运动。小球以某个初速度从地面跳起，再次落地弹起后速度减至原来的 90%，计算它经过 10 次弹跳的运动轨迹并用图形显示，在 MATLAB5.3 中程序如下：

```
function varargout = ballode(t,y,flag)
%BALLODE Equations of motion for a bouncing ball.
% BALLODE(T,Y) returns the derivatives vector for the equations of
% motion for a bouncing ball.  This ODE file can be used to test the
% zero-crossing location capabilities of the ODE Suite solvers.
%
% BALLODE(T,Y,'events') returns a zero-crossing vector VALUE evaluated
% at (T,Y) and two constant vectors ISTERMINAL and DIRECTION.By default,
% the solvers of the ODE Suite do not locate zero-crossings.However,if
% the ODE solver property Events is set to 'on' with ODESET, the solver
% calls the ODE file with the flag 'events'.  The ODE file returns the
% information that the solver uses to locate zero-crossings of the
% elements in the VALUE vector.  The VALUE vector may be of any length.
% It is evaluated at the beginning and end of a step,and if any elements
% make transitions to, from, or through zero (with the directionality
% specified in DIRECTION), then the zero-crossing point is located.The
% ISTERMINAL vector consists of logical 1's and 0's, enabling you to
```

```
% specify whether or not a zero-crossing of the corresponding VALUE
% element halts the integration.  The DIRECTION vector enables you to
% specify a desired directionality, positive (1), negative (-1), and
% don't care (0) for each VALUE element.
%
% BALLODE(T,Y,'init') returns initial conditions (see RIGIDODE).
%
%  BALLODE with no input arguments runs a demo of a bouncing ball.It is
%  an example of repeated event location, where the initial conditions
%  are changed after each terminal event.  This demo computes ten
%  bounces with calls to ODE45.  The speed of the ball is attenuated by
%  0.9 after each bounce, and the simulation is stopped when the time
%  for a bounce has decreased to a minimum length.  Note that the event
%  function also locates the peak of each bounce.
%
%  See also ODE45, ODESET, ODEFILE.

%  Mark W. Reichelt and Lawrence F. Shampine, 1/3/95
%  Copyright (c) 1984-98 by The MathWorks, Inc.
%  $Revision: 1.10 $  $Date: 1997/11/21 23:25:02 $

if nargin == 0
flag = 'demo';
end

switch flag
case  ''                          % Return dy/dt = f(t,y).
    varargout{1} = f(t,y);
case  'init'                      % Return default [tspan,y0,options].
    [varargout{1:3}] = init;
case  'events'                    % Return [value,isterminal,direction].
    [varargout{1:3}] = events(t,y);
case  'demo'                      % Run a demo.
    demo;
otherwise
```

```
    error(['Unknown flag ''' flag '''.']);
end

% ----------------------------------------------------------------------

function dydt = f(t,y)
dydt = [y(2); -9.8];

% ----------------------------------------------------------------------

function [tspan,y0,options] = init
tspan = [0; 10];
y0 = [0; 20];
options = odeset('Events','on');

% ----------------------------------------------------------------------

function [value,isterminal,direction] = events(t,y)
% Locate the time when height passes through zero in a decreasing
% direction and stop integration.  Also locate both decreasing and
% increasing zero-crossings of velocity, and don't stop integration.
value = y;                              % [height; velocity]
isterminal = [1; 0];
direction = [-1; 0];

% ----------------------------------------------------------------------

function demo

tstart = 0;
tfinal = 30;
y0 = [0; 20];
refine = 4;
options = odeset('Events','on','OutputFcn','odeplot','OutputSel',1,...
                'Refine',refine);

clf reset                             % deletes any stop button
set(gca,'xlim',[0 30],'ylim',[0 25]);
box on
hold on;
```

```
tout = tstart;
yout = y0.';
teout = [ ];
yeout = [ ];
ieout = [ ];
for i = 1:10
  % Solve until the first terminal event.
  [t,y,te,ye,ie] = ode23('ballode',[tstart tfinal],y0,options);
  if  ishold
    hold on
  end
  % Accumulate output.  This could be passed out as output arguments.
  nt = length(t);
  tout = [tout; t(2:nt)];
  yout = [yout; y(2:nt,:)];
  teout = [teout; te];     % Events at tstart are never reported.
  yeout = [yeout; ye];
  ieout = [ieout; ie];

  ud = get(gcf,'UserData');
  if ud.stop
    break;
  end

  % Set the new initial conditions, with .9 attenuation.
  y0(1) = 0;
  y0(2) = -.9*y(nt,2);

  % A good guess of a valid first timestep is the length of the last
  % valid timestep, so use it for faster computation.  'refine'  is
  % 4 by default.
  options = odeset(options,'InitialStep',t(nt)-t(nt-refine),...
                  'MaxStep',t(nt)-t(1));

  tstart = t(nt);
end
```

这是一个在 MATLAB 5.3 内部的演示程序，是完全按照模板的格式书写的一个较复杂的程序，同时它也体现了一般的程序文件的一些基本特征。从这个例子中可以更多地了解模板的用法。读者可以用这种方法去查看 MATLAB 的其它程序。

调用程序的方法是，键入

```
>> ballode
```

或者键入

```
>> ballode(T,Y,'demo')
```

前一种格式是以文件中的默认值来演示程序，后一种格式是由用户给定的解方程的时间范围 T 和初始条件 Y 来演示程序，其中 demo 则是该文件中 flag 的赋值。

原文件中有英文说明，为了便于读者理解，下面加以简单说明。

程序中的 flag 有五种选项，比例 1 增加了两个选项，即 'init' 和 'events'，前者是输出解方程的初始条件；后者是在程序中用 options 将 'events' 选项设为 'on'，用它来控制积分中断的时刻。而 'events' 值从子函数文件 events 中输出，设置的办法是：在文件中，当 isterminal 取 1 时表示中断积分，取 0 则不中断积分，而 direction 表示变化的方向，取 1 表示判断事件的变量从负值增加到零，取 −1 则从正值减小到零，取 0 则不考虑方向。 y 的两个分量分别是高度和速度，判断事件的变量是高度，程序中的设置是当高度减小到零时，积分中止。然后在主程序中重新给出新的初始条件，开始新的一次计算。

在 MATLAB6.0 中，由于使用了函数句柄，这个程序已经简化为如下程序：

```
function  ballode
% BALLODE Run a demo of a bouncing ball.
%   This is an example of repeated event location, where the
%   initial conditions are changed after each terminal event.  This
%   demo computes ten bounces with calls to ODE23.  The speed of the
%   ball is attenuated by 0.9 after each bounce.  The trajectory is
%   plotted using the output function ODEPLOT.
%
%   See also ODE23, ODE45, ODESET, ODEPLOT, @.
%   Mark W. Reichelt and Lawrence F. Shampine, 1/3/95
%   Copyright 1984-2001 The MathWorks, Inc.
%   $ Revision:  1.16 $ $Date:  2001/04/15 12:02:54 $

tstart = 0;
tfinal = 30;
```

```
y0 = [0; 20];
refine = 4;
options = odeset('Events',@events,'OutputFcn',@odeplot,'OutputSel',1,...
             'Refine',refine);

figure;
set(gca, 'xlim', [0 30], 'ylim', [0 25]);
box on
hold on;
tout = tstart;
yout = y0.';
teout = [ ];
yeout = [ ];
ieout = [ ];
for i = 1:10
    % Solve until the first terminal event.
    [t,y,te,ye,ie] = ode23(@f,[tstart tfinal],y0,options);
    if ~ishold
      hold on
    end
    % Accumulate output.  This could be passed out as output arguments.
    nt = length(t);
    tout = [tout; t(2:nt)];
    yout = [yout; y(2:nt,:)];
    teout = [teout; te];    % Events at tstart are never reported.
    yeout = [yeout; ye];
    ieout = [ieout; ie];

    ud = get(gcf, 'UserData');
    if ud.stop
      break;
    end

    % Set the new initial conditions, with .9 attenuation.
    y0(1) = 0;
    y0(2) = -.9*y(nt,2);
```

```
    % A good guess of a valid first timestep is the length of the
    % last valid timestep, so use it for faster computation.
    % 'refine' is 4 bydefault.
    options = odeset(options,'InitialStep',t(nt)-t(nt-refine),...
                'MaxStep',t(nt)-t(1));

    tstart = t(nt);
end

plot(teout,yeout(:,1),'ro')
xlabel('time');
ylabel('height');
title('Ball trajectory and the events');
hold off
odeplot([ ],[ ],'done');

% ---------------------------------------------------------------------

function dydt = f(t,y)
dydt = [y(2); -9.8];

% ---------------------------------------------------------------------

function [value,isterminal,direction] = events(t,y)
% Locate the time when height passes through zero in a decreasing
% direction and stop integration.
value = y(1);      % detect height = 0
isterminal = 1;    % stop the integration
direction = -1;    % negative direction
```

从这个演示程序的变化，读者可以体会到，MATLAB 总在不断地改进自身的性能，它的确是一个优秀的计算软件。

1.5 作图和动画

无论在科研或教学上，计算的结果都希望能做到可视化，而 MATLAB 完善与强大的作图功能，正好满足了这种需求。

MATLAB 可以画二维图和三维图；可以在二维图上用颜色表示三维数据，也可以在三维图上用切片和颜色来表示四维图形。可以画三维表面图形，也可

以画三维的立体图，还可以用矢量数据画三维流线场、三维矢量场、三维表面的法线等等。可以用数据直接作图，也可以用函数画图，有一些用字母 ez 开头的作图指令还可以对隐函数和参数函数作图。 MATLAB 可以用不同的方法制作动画，还可以制作用户界面，使得编成的程序更为方便实用。对一些特殊的作图要求，如对数图、复数图、极坐标图、矢量场图、等高线图，在图形上加上文字注释和改变图形的视角等，它都有直接的指令。利用图形窗口的编辑功能可以直接对图形进行操作，如增加标题和说明性文字、改变视角、画线条和箭头、将图形放大缩小、改变线型、旋转三维图形等等；而利用图形属性编辑器 (Graphics Property Editor) 可以对图形对象的各种属性如颜色、位置等进行更全面的设置。所得的图形不仅可以用 MATLAB 自身默认的图形文件格式存盘或打印输出，更可以用多达十数种图形文件格式如 eps, bmp, jpg, tif 等输出，这些都是科研工作者投稿常用的图形文件格式，而为了实现这些任务用户只要在图形窗口中的菜单 File 下选择 Export ，再找到相应的文件格式就行。

1.5.1 二维图形

1. 创建图形

最基本的作图指令是 plot ，它有不同的形式，与输入的量有关。对于矢量 y ， plot(y) 产生一个折线图，纵轴为 y 的元素，横轴为 y 的元素指标。如果输入两个矢量 x, y, 则 plot(x, y) 产生的是 y 相对于 x 的图形。画出 0 到 π 的正弦图形的命令是

```
>> t = 0 : pi/100 : 2*pi ;
>> y = sin(t);
>> plot(t, y)
```

一个 plot 命令可以同时画多个图形。 MATLAB 会自动地用不同的颜色区分每组数据的图形，颜色是预先设置好的，但可以改动。如下面的命令用不同的颜色在一个坐标系内画出了两个相对于 t 的函数，

```
>> t = 0:0.1:2*pi;
>> y1 = sin(t-.25);  y2 = sin(t-1);
>> plot(t, y1, t, y2)
```

指定颜色、线型和数据点标志的命令格式如下：

plot(x,y,'color style marker')

例如 plot(x, y, 'y : + ') 画出的是黄色点状线，在每个数据点有加号标志。如果指定了数据点标志而不指定线型，则只在各个数据点画出标志符号。颜色、线型和数据点标志的可选项有

颜色：青 'c' ，洋红 'm' ，黄 'y' ，红 'r' ，绿 'g' ，蓝 'b' ，白 'w' ，黑 'k'

线型：实线 '−' ，短划线 '−−'，点线 ':'，点划线 '−.'

标志：点 '·', 加号 '+', 圆圈 'o', 星号 '*', 叉号 '×', 四方形 's', 钻石形 'd', 五角形 'p', 六角形 'h', 三角形 (向上 'v', 向下 '∧', 向左 '<', 向右 '>')

2. 图形窗口

每次作图，作图命令都会自动打开一个图形窗口，但是如果已经有图形窗口存在，作图命令便会使用已有的图形窗口。如果键入 figure，就会打开一个新的图形窗口。每个图形窗口的标志栏都会有一个编号 n，打开第 n 个图形窗口的指令是

```
figure(n)
```

在已有图形上继续作图的指令是

```
hold on
```

取消这种功能的指令是

```
hold off
```

打开的图形窗口如图 1.13 所示

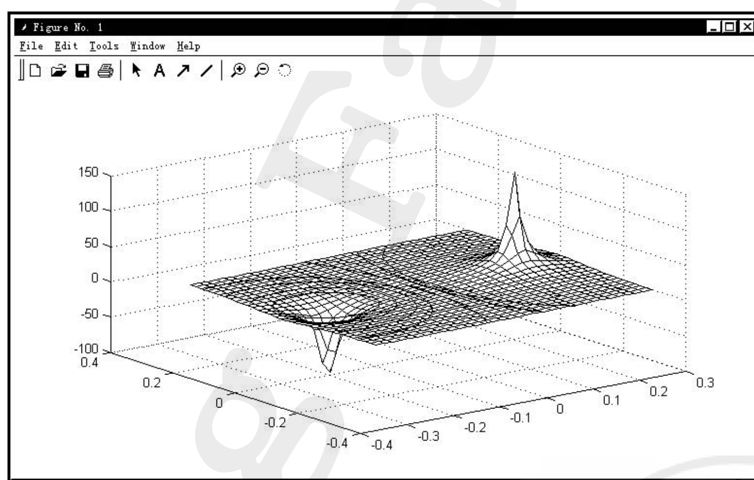


图 1.13 电偶极子的势能

在图中，上端是菜单栏，选择 Tools 栏的 Show Toolbar 可以显示工具栏的图标。各图标的含义分别是打开、关闭、保存和打印文件，编辑图形，加注文字，画箭头和画直线，放大、缩小和旋转图形。

直接点击图形中的曲线，会打开一个对话框。可以对曲线的各种属性如线宽、颜色、标志等加以编辑。File 菜单下的选项 (Preferences) 可以改变图形窗口的各种功能，如数据的格式、字体、图形存储的格式等。

下面的程序文件 st.m 先画一个电偶极子的势能分布图，然后再叠加同一个

函数的等高线图。用图形窗口的各种功能可以加注解、改变视角等。程序画出的图形如图 1.13 所示。

```
x=-0.3:0.018:0.3;  y=-0.3:0.018:0.3;
[X,Y]=meshgrid(x,y);       %%数据网格
Z=1./sqrt((X-0.2).^2.+Y.^2)-1./sqrt((X+0.2).^2+Y.^2);   %%势能函数
mesh(x,y,Z)      %%势能的三维表面网格图
hold on
contour(x,y,Z,[-8,-5,-3,-2,1,-0.5,0.5,1,2,3,5,8],'k')    %%等高线图
```

3. 分区作图

要在一个窗口画几个图，可以用分区作图指令

subplot(m,n,p)

它将窗口分成 m × n 个区，再选择第 p 个区作为当前活动窗口作图。图形按照窗口的第一行、第二行依次编号。例如下面的程序 qtzt.m 画出了图 1.14，它在一个窗口中画出了两个旋转体的表面，所用到的指令在后面介绍三维图形时都会讲到。

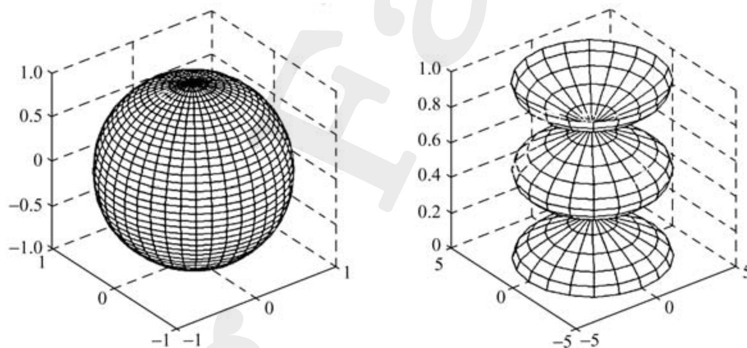


图 1.14 分区作图的例子

```
%% program qtzt.m
subplot(1,2,1)
sphere(40)
axis equal
subplot(1,2,2)
t=0:0.1*pi:2*pi;
[X,Y,Z]=cylinder(4*cos(t));
mesh(X,Y,Z)
```

```
axis square
```

4. 复数作图

用复数作图，一般是将虚部丢弃。如果对单个复数作图，则作图指令用实部对虚部作图。因此 plot(Z) 是表示 plot(real(Z), imag(Z))，例如

```
>> t = 0 : pi/10 : 2*pi;
>> plot(exp(i*t), '- o')
```

画出了一个 20 条边的多边形，每个顶点有一个小圆圈。

对于以复数为变量作复函数图形有专门的命令，如绘制复变量函数图形的命令 cplxmap，绘制复数第 n 个根的黎曼表面图的命令 cplxroot。

5. 用函数画二维曲线

利用函数直接绘制二维曲线的命令是 fplot，它绘图的数据点是自适应产生的，即在函数变化小的地方取较少的点，在函数变化剧烈的地方取较多的点。如

```
>> fplot('sin(1./x)', [0.01 0.1], 1e-3)
```

直接画出了在 [0.01 0.1] 区间内函数 $\sin(1/x)$ 的曲线，其误差为 10^{-3}。实际图形如图 1.15 所示。

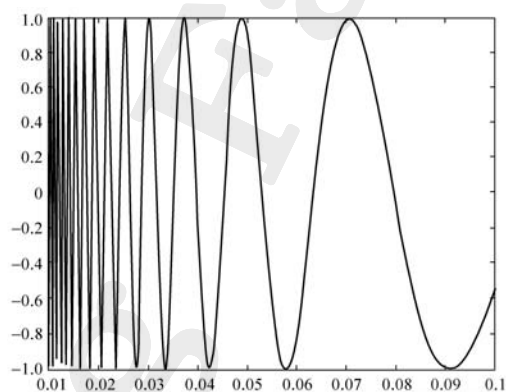


图 1.15　自适应法绘制的图形

图形的左边相当密集，为了放大图形，可以用图形窗口里菜单 Tools 下的指令 zoom in，或者直接点击图标栏中表示放大的图标，然后再用鼠标指到想查看的地方，按下左键，在屏幕上画出一个矩形框，再松开左键，便可将框内图形放大。由于是自适应作图，所以仍能看到形状比较正确的曲线。如想恢复原图大小，可选择指令 zoom out 或点击相应的图标以缩小图形。

还有一些指令和 fplot 用法相似，也可以用函数直接画二维或三维图，如

ezplot, ezplot3， ezmesh， ezpolar 等等，它们不仅具有与 fplot 相似的功能，还可以对隐函数作图。这些指令都收录在附录 A20.7 中。

6. 二维特殊图形

MATLAB 有许多画特殊图形的专门指令（见附录 A20.7），点击 Help 菜单下的 Examples and Demos 菜单，在对话框中选择 Visualization，再在右边的对话框中选择 2-D Plots 就可以演示各种二维作图指令。如果选择其它选项，则可以演示其它作图指令。下面列举了一些画二维图的指令：

bar	直方图	loglog	双对数坐标曲线
compass	原点出发的复数矢量图	pcolor	假彩色图
contour	在 x - y 面上的等高线图	polar	极坐标图
errobar	误差棒图	quiver	矢量场图
ezplot	符号函数二维曲线	rose	统计频数扇块图
feather	沿 x 轴分布的复数矢量图	semilogx	x 轴对数坐标曲线
fplot	数值函数二维曲线	semilogy	y 轴对数坐标曲线
fill	平面多边形填色	stem	火柴杆图
hist	统计频数直方图	stairs	阶梯图

7. 坐标轴控制

指令 axis 有许多控制坐标轴的选项，用以确定伸缩比率、方向及图形外观比例。一般地， MATLAB 会自动地寻找数据的最大值和最小值，选择合适的作图范围和坐标标度。指令 axis 可以不受默认设置的限制。如

axis([xmin xmax ymin ymax])	x 轴和 y 轴的取值范围；
axis square	正方形的 x 轴和 y 轴；
axis equal	在 x,y 轴上画出等长的刻度线；
axis auto	把坐标轴的伸缩率设置为自动模式；
axis on	打开坐标轴的标签和刻度线；
axis off	取消坐标轴的标签和刻度线；
grid off	取消网格线；
grid on	恢复网格线；
xlabel, ylabel, zlabel	加上 x 轴， y 轴和 z 轴的标注；
title	在图的上端增加一个标题；
text	将文字插入图中。

注意，在图中插入的希腊文字母和数学符号是用 LaTex 符号产生，也可以用微软拼音输入法里的软键盘产生。

如在图 1.14 中就使用了指令 axis equal 和 axis squal，读者可以看看取消两条指令以后的运行效果。

下面的程序 zbz.m 画出了图 1.16，图中对坐标轴进行了标注，并在图形上加了文字说明，其中 \ leq 表示小于， \ pi 表示 π， \it 表示斜体。当然，对坐标轴控制和标注也可以由图形窗口的有关操作来完成。

```
%% program zbz.m
t = -pi : pi/100 : pi;
y = sin(t);
plot(t, y)
axis([-pi pi -1 1])
xlabel('-\pi \leq {\it t} \leq \pi', 'fontsize', 20)
ylabel('sin(t)', 'fontsize', 20)
title('Graph of the sine function', 'fontsize', 15)
text(-2.5, 1/2, '{\it Note the odd symmetry.}', 'fontsize', 20)
```

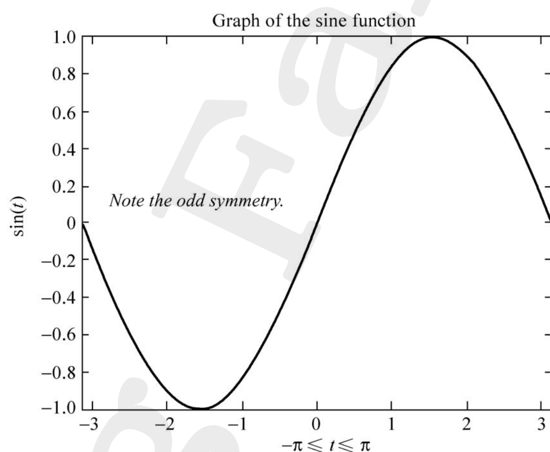


图 1.16　加了文字标注的图形

1.5.2　三维图形

三维图形分三维曲线、三维曲面和立体图形。它们的作图指令不同。

1. 三维空间曲线

三维曲线的作图指令为

plot3(x, y, z, s)

这里 x, y, z 是同维数的矢量或矩阵。每组 x, y, z 构成一个点的坐标，各点依次相联，形成一条曲线。如果是矩阵，则它们相应的列构成一条三维曲线的数据点坐标，所以用矩阵可以同时画多条空间曲线。 s 是线型、颜色和标志的参数。

下面的程序 kjqx.m 画出了图 1.17 中的空间曲线和四面体。空间曲线是用参数方程画的。四面体则是利用 plot3 将空间的点连接起来而形成的空间的几何图形，其中数据 x1, y1, z1 代表左面竖直的三角形，数据 x2, y2, z2 代表右面竖直的三角形，而数据 x3, y3, z3 代表在前面下方坐标轴上的横线。画好以后用指令 view 选择适当的视角，并且在两个表面填上不同的颜色。

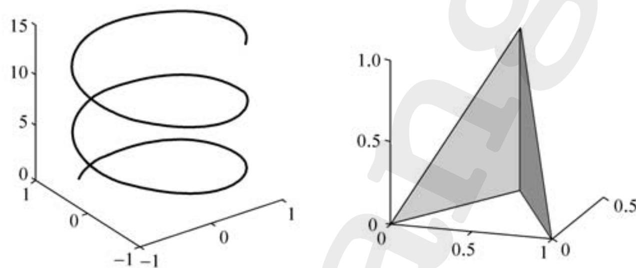


图 1.17 空间曲线和空间四面体

```
%%program kjqx.m
subplot(1, 2, 1)
t = 0 : 0.01 : 15;
x = sin(t);  y = cos(t);  z = t;
plot3(x, y, z)
view(-60, 18)

subplot(1,2,2)
x1=[0.5,  0.5,  0,  0.5];
y1=[0.5,  0.5,  0,  0.5];
z1=[  1,    0,  0,    1];
x2=[0.5,  0.5,  1,  0.5];
y2=[0.5,  0.5,  0,  0.5];
z2=[  1,    0,  0,    1];
x3=[0, 1];
y3=[0, 0];
z3=[0, 0];
```

```
plot3(x1,y1,z1, x2,y2,z2, x3,y3,z3)
hold on
fill3(x1, y1, z1, 'g')
fill3(x2, y2, z2, 'r')
view(24, 30)
```

MATLAB 填色的指令对二维图形是 fill, 对三维图形是 fill3 。上面用 fill3 对四面体竖直的两个表面分别填上了红色和绿色。指令 patch 兼有这两者的功能而且功能更多。上述最后两条语句也可改成

```
patch(x1, y1, z1, 'g')
patch(x2, y2, z2, 'r')
```

更复杂的图形也是这样画，只是数学关系的表述更复杂。下面的程序 qzb.m 画的是球坐标系中的一个小体元，结果如图 1.18 所示。

```
%% program qzb.m
axis([0 1 0 1 0 1])
xlabel('x'); ylabel('y'); zlabel('z')
hold on
plot3([0 1],[0 0],[0 0],'k',[0 0],[0 1],[0 0],'k',...
      [0 0],[0 0],[0 1],'k')
view(110,15)

%% 画半径
r=[0,1]; a=pi/4; b=pi/4;
[x1,y1,z1]=sph2cart([0,a],[0,b],r);
[x2,y2,z2]=sph2cart([0,a+pi*0.05],[0,b],r);
[x3,y3,z3]=sph2cart([0,a],[0,b+pi*0.05],r);
[x4,y4,z4]=sph2cart([0,a+pi*0.05],[0,b+pi*0.05],r);
plot3(x1,y1,z1,'r',x2,y2,z2,'r', x3,y3,z3,'r',x4,y4,z4,'m :' )

%% 画小体元外表面
r1=ones(1,11);
theta1=a*ones(1,11);
phi1=b:0.005*pi:b+pi*0.05;
[x1,y1,z1]=sph2cart(theta1,phi1,r1);
theta2=(a+pi*0.05)*ones(1,11);
phi2=b:0.005*pi:b+pi*0.05;
[x2,y2,z2]=sph2cart(theta2,phi2,r1);
```

```
theta3=a:0.005*pi:a+pi*0.05;
phi3=b*ones(1,11);
[x3,y3,z3]=sph2cart(theta3,phi3,r1);
theta4=a:0.005*pi:a+pi*0.05;
phi4=(b+pi*0.05)*ones(1,11);
[x4,y4,z4]=sph2cart(theta4,phi4,r1);
plot3(x1,y1,z1,'b',x2,y2,z2,'b', x3,y3,z3,'b',x4,y4,z4,'b')

%% 画小体元内表面
r2=0.85*r1;
[x5,y5,z5]=sph2cart(theta1,phi1,r2);
[x6,y6,z6]=sph2cart(theta2,phi2,r2);
[x7,y7,z7]=sph2cart(theta3,phi3,r2);
[x8,y8,z8]=sph2cart(theta4,phi4,r2);
plot3(x5,y5,z5,'b',x6,y6,z6,'c :', x7,y7,z7,'b',x8,y8,z8,'c :')
```

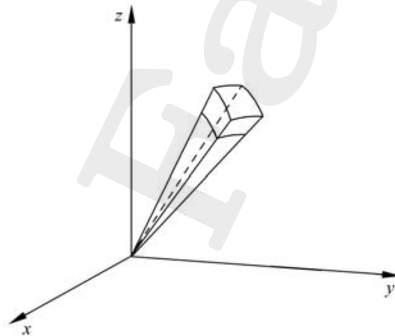


图 1.18 球坐标系中的小体积元

2. 画球面、椭球面和柱面

画球面、椭球面和柱面有专门的指令。 sphere(N) 是画一个半径为 1，有 N 条经线的球面。 cylinder(R,N) 是画由母线 R 旋转产生的柱面，在柱面上画有 N 条母线。图 1.14 就运用了这两个指令。 ellipsoid(XC, YC, ZC, XR, YR, ZR, N) 是画椭球面，描述该椭球面的方程是

$$\frac{(X-XC)^2}{XR^2}+\frac{(Y-YC)^2}{YR^2}+\frac{(Z-ZC)^2}{ZR^2}=1$$

也可以用这些指令生成相应的网格数据，如 [X, Y, Z]=cylinder(R, N)，有了这些数据网格就可以用下面介绍的三维作图指令去作图，图 1.14 就分别使用了这两

种不同的方法。

3. 网线和表面图

在 MATLAB 中，曲面是用 xy 平面上的各个格点的 z 坐标来定义，相邻点用直线连接。指令 mesh 和 surf 都是用三维方式显示曲面。 mesh 产生网线状的曲面，只有定义点的连线有颜色。 surf 则用颜色显示连线及其之间的面积。此外，也可以用 meshc 和 meshz 来绘制网线图，其中 meshc 是将 mesh 图和等高线图绘制在一起，而 meshz 则是在绘制 mesh 图时画出零基准平面。如果要绘制瀑布线图，可用指令 waterfall 。在 demos 的演示中有许多例子。

为了显示两变量的函数 z=f(x, y), 要在函数域上产生一个数据网格 X, Y, 然后用 X, Y 来计算和作图。指令 meshgrid 将矢量 x, y 决定的函数域变换为数据网格 X, Y 。 X 的每一列都是矢量 x, Y 的每一行都是矢量 y, 读者可以复习前面 1.2 节对数据网格的说明。 Z 是两变量函数的值。最后用 mesh(X, Y, Z) 作图。如果作图指令是 mesh(Z), 则代表 X, Y 坐标的是 Z 的列标和行标。具体的例子可见图 1.13 。读者可以将指令 mesh 改换成其它画三维图形的指令，然后重新运行程序，比较它们所得的结果。

4. 视角

如果要改变观察三维图形的视角，可使用命令

view(az, el)

其中 az 表示方位角，单位是度，取值为 $-180 \sim 180$ ，计算起点是负 y 轴， el 表示俯视角，取值为 $-90 \sim 90$ ，计算起点是 xy 平面。进行三维观察时的默认值是 az=−37.5, el=30 , 进行二维观察时的默认值是 az=0, el=90 。在图形窗口中，用鼠标在工具栏中选择旋转图形的图标，再将鼠标指向图形，就会显示当前的视角。

5. 色彩的调制、渲染和光照的控制

图形的色彩可以用指令

colormap（ MAP ）

来设置和改变， MAP 是色图函数或色图矩阵。附录 A20.11 列出了所有的色图函数。色图矩阵是 3 列多行的矩阵，它的每一行代表一种颜色，每一行中的三个元素分别表示这种颜色中红色、绿色和蓝色的强度，强度值变化范围为 0.0 到 1.0 。下表是色图矩阵中行矢量与色彩的对应关系。

[0 0 0]	黑色	[1 0 1]	品红色
[1 0 0]	纯红	[0.5 0 0]	暗红色
[0 0 1]	蓝色	[0.5 0.5 0.5]	灰色

[1 1 1]	白色	[1 0.62 0.40]	纯铜色
[0 1 1]	青色	[0.49 1 0.48]	宝石蓝
[0 1 0]	绿色	[127/255 1 212/255]	浅绿色
[1 1 0]	黄色		

对由 surf, mesh, pcolor, fill, fill3 所创建的图形，可用指令 shading 进行色彩渲染。渲染的方式有

shading flat 平坦式渲染，即用线段两端点或小方块面的四角的色值决定线段和方块面的颜色；

shading interp 插补式渲染，即线段或小方块面的颜色是以端点值进行插值而得到线性变化的颜色；

shading faceted 默认方式，是平坦式渲染加上黑色网格线。

控制光照效果的指令是

surfl(X,Y,Z,S,K)

与指令 surf 相比，它有两个控制光照效果的参数 S, K 。 S 用以确定光源的位置，可用直角坐标 S=[sx, sy, sz] 表示，也可用球坐标中的方位角和俯视角 S = [az, el] 表示，默认的光源方位角是在观察点逆时针方向 45° 处。 K 是用光方式， K=[背景光份额，漫射光份额，定向光份额，扩散系数] 。下面的程序 ts.m 在四个图中使用了色彩调制以及不同的色彩渲染和光照效果，使用图形窗口的图标转动图形，可以更明显地看出它们之间的差别。

```
[X,Y] = meshgrid(-8:.5:8);
R = sqrt(X.^2 + Y.^2) + eps;
Z = sin(R)./R;
colormap(pink)
subplot(2,2,1)
surfl(X,Y,Z)
subplot(2,2,2)
surfl(X,Y,Z,[40,60],[0.1,0.1,0.7,1])
subplot(2,2,3)
surfl(X,Y,Z)
shading flat
subplot(2,2,4)
surfl(X,Y,Z,[40,60],[0.1,0.1,0.7,1])
shading interp
```

6. 四维数据的表现

三元函数会产生四维数据，MATLAB 表现三元函数的办法是对三维物体切片，然后在切面上用颜色表现数据。颜色与数据的对应关系则由色轴来表示。下面的程序 ys.m 画出了图 1.19 。为了表现函数

$$v = xe^{-x^2-y^2-z^2}$$

的值，利用指令 slice 作了四个截面，分别是 x=5 ， x=15 ， y=15 ， z=10 ；每个截面上的函数值用颜色来表示，颜色与数值的对应关系由色轴来表示，色轴由指令 colorbar 来画出，图中画了垂直和水平两条色轴。从图中可以看出，函数值有一个峰和一个谷，实际上，这是一个三变量的高斯型函数与变量 x 的乘积，根据切片的颜色，读者可以很容易地想象出函数值在空间的分布。

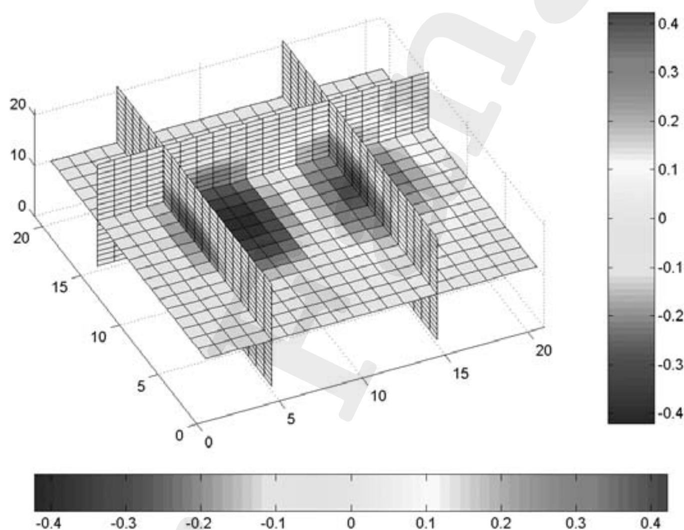


图 1.19　用切片颜色表示四维数据

```
%%  program  ys.m
[x,y,z] = meshgrid(-2:.2:2,-2:.2:2,-2:.2:2);
v = x .* exp(-x.^2 - y.^2 - z.^2);
slice(v,[5 15],15,10)
axis([0 21 0 21 0 21]);
hold on
colorbar('horiz')
colorbar('vert')
```

```
view([-25 65])
```

1.5.3 句柄图形

假定我们要制作一幅实时动画来表现小球在桌面上的跳动。做法是擦除旧的小球图像，在新的位置画出一个小球的图像。在 MATLAB 中，完成这个任务就运用了句柄图像。具体来说，就是把小球看作一个图像对象，首先要获得对小球图像的控制权，也就是获得这个对象的句柄，然后不断改变这个对象的位置属性就能达到这个目的。在这个操作中，我们操作的对象不是整个图形，而只是图形中的一部分。由此可见，句柄图形系统是一种面向对象（按照对象进行操作）的图形系统。

1. 句柄图形的结构层次

句柄图形系统包含以下含义：一幅图的各个组成部分都是一个对象，每一个对象有一系列句柄来标识它，每一个对象都有可以设置和改变的属性。

图形对象按父对象和子对象组成层次结构。计算机屏幕是根对象，并且是所有其它对象的父对象。图形窗口是根对象的子对象，坐标轴的用户界面等对象是图形窗口的子对象。线条、文本、表面和图像等对象是坐标轴对象的子对象。这种层次关系如图 1.20 所示。一个父对象可以包含一个或多个子对象，如根屏幕可包含一个或多个图形窗口。



图 1.20 句柄图形系统的层次关系

2. 访问图形对象句柄

句柄实际上就是分配给每个对象的唯一的数字标识，它是在创建图形对象时自动建立的。访问图形对象句柄的指令有

gcf	获取当前图形窗口的句柄
gca	获取当前坐标轴的句柄
gco	获取当前对象的句柄
gcbo	获取当前正在调用的对象的句柄
gcbf	获取包括当前正在调用的对象的图形的句柄

findobj 查找具有某一属性的图形对象的句柄
copyobj 复制图形对象
delete 删除图形对象

3. 设置图形对象属性

线条有粗细，字符的字体有大小，这些都叫做图形对象的属性。图形对象的属性包括属性名（字符串）和相应的数值，改变图形对象的某个属性的值就能改变图形对象。

(1) get 获取图形对象的属性值

语句格式为

PropertyValue = get(h)

PropertyValue = get(H, 'PropertyName')

第一种格式的含义为获取图形句柄为 h 的图像属性，操作的结果是返回一个结构数组，结构数组的各个域名和赋值对应图形对象的各个属性和属性值。第二种格式的含义为获取图形句柄为 H 的图像属性名为 PropertyName 的属性值。

(2) set 设置图形对象的属性值

语句格式为

set(H, '属性名', '属性值')

set(H, '属性名 1', '属性值 1', '属性名 2', '属性值 2', ⋯)

第一种格式是对句柄为 H 的图形对象中指定属性名的属性值加以设置。第二种格式则可以同时设置多个属性值。

1.5.4 动画

用动画表现计算结果可以显得生动逼真。MATLAB 有多种动画表现方式，分述如下：

1. 彗星轨迹图

知道点的位置数据，可以用彗星轨迹图动态地显示质点的运动轨迹，其指令为

comet(x, y, p) 彗长为 p*length(y) 的二维彗星轨迹， p 的默认值为 0.1 。

comet3(x, y, z, p) 彗长为 p*length(y) 的三维彗星轨迹， p 的默认值为 0.1 。

2. 转动图形

在图形窗口中，利用菜单 Tools 下的 Rotate 3D 可以用鼠标对图形作三维转动，在程序中可用指令 rotate3d 来完成这个任务。如果要使图形自动地连续转动，所用的指令是 rotate 。其指令格式为：

rotate(H, [theta phi], alpha)

rotate(H, [x y z], alpha)

rotate(··· , origin)

各项符号的含义为：

H	图形对象的句柄；
theta, phi	以度为单位的球坐标中的方位角和极角，表示转轴的方向；
alpha	从原来位置绕转轴转动的角度，正值是按右手螺旋法则计算；
x, y, z	用直角坐标 (x, y, z) 表示转轴的终点，起点是坐标轴原点；
origin	用 [x0, y0, z0] 作为直角坐标的原点。

3. 影片动画

制作影片动画与电影制作很相似，即预先将一幅一幅的图形制作好，放在内存缓冲区内，然后连续地播放，播放的速度可快可慢，这种方法需要的内存较大。它有三个步骤，分别由三个指令 moviein, getframe 和 movie 来完成。

M=moviein(n)，定义一个 n 列的矩阵 M，以存储 n 帧画面，每列对应一帧。

M(:, j) = getframe，这步是依次将 n 帧画面存入矩阵 M，通常用 for 循环来完成它。

movie(M, k, FPS)，这步是将 n 帧画面演示 k 次，每秒 FPS 帧。 k 默认为一次， FPS 默认为 12 。

4. 实时动画

在实时动画的制作中，对没有改变的背景图案不作更新，只更新运动部分的图案，所以它加快了每幅图的生成速度，也节省了内存。计算机游戏都采用这种方法。

MATLAB 用图形对象的 'EraseMode' 属性来实现在保持背景图案的条件下擦除旧对象，显示新对象。图形对象的 'EraseMode' 属性有四种：

normal	重绘整个显示区，它产生的图形准确但速度慢；
none	不做任何擦除，直接在原图上绘图；
xor	擦除旧对象的点并绘制新对象的点；
background	把旧对象的颜色变为背景色。

当新对象的属性修改以后，应该用指令 drawnow 刷新屏幕，使新对象显示出来。指令 drawnow 使 MATLAB 暂停目前的任务而去刷新屏幕，若不使用 drawnow, 则要等到任务序列执行完毕以后才会刷新屏幕。

下面的程序是行波的演示。程序 wave1.m 是用影片动画制作的，程序 wave2.m 是用实时动画制作的。在第二个程序中， h=plot(x,y) 是获得图形的句柄， set 语句则是对图形属性重新设置。

```
%wave1.m
t=0:pi/20:4*pi;
x=0:0.1*pi:4*pi;
m=moviein(40)
for i=1:40
   y=sin(x-t(i));
   plot(x,y);
   axis([0,12,-1,1]);
   m(:,i)=getframe;
end
movie(m,3,30)

%wave2.m
t=0:pi/20:4*pi;
x=0:0.1*pi:4*pi;
h=plot(x,y);
axis([0,12,-1,1]);
set(h,'EraseMode','xor')
for i=2:60
   y=sin(x-t(i));
   set(h,'XData',x,'YData',y)
   drawnow;
end
```

5. 打印和输出图形

可以用图形窗口指令 print 来直接打印屏幕上的图像；也可以用菜单 print 先设置标准的打印选项后再打印；还可以将图形存储在专门的文件中并选用不同的输出格式，包括 eps, jpg, bmp 等格式，做法是依次选取菜单 File\Export 项，就会打开对话框，然后选取所要的格式。

1.6 工具箱

1.6.1 符号运算工具箱

MATLAB 的符号运算功能是借助于加拿大滑铁卢大学研制的数学软件 MAPLE 的核心功能开发的。它包括两个工具箱：基本的符号运算工具箱 (Symbolic

Math Toolbox) 和扩展的符号运算工具箱 (Extended Symbolic Math Toolbox) 。

基本的符号工具箱提供了 100 多条指令来使用 MAPLE 的线性代数工具包的功能，本书附录 B 列出了这些指令。这些指令是根据 MATLAB 的语法和风格编写的，对熟悉 MATLAB 的人来说，可以像使用其它的指令一样方便地使用它们。扩展的符号工具箱大大地增强了使用 MAPLE 的能力，你可以进入所有的 MAPLE 工具包 (图形工具包除外)，使用 MAPLE 的程序以及用户定义的程序。扩展的符号工具箱提供的是对 MAPLE 函数的接口，然后以指定的格式调用 MAPLE 自身的指令，运用这种方式作符号计算要求掌握 MAPLE 的基本语言。

本书附录 B 的《符号工具箱指令》中第 12 项是符号工具箱基本功能的演示指令，读者可以通过它们对符号工具箱的用法获得一个初步了解。在指令窗口键入

```
>> help symbolic
```

可以查看符号运算工具箱的全部指令及其简单说明。

在 1.2.3 节已经介绍了符号变量的定义，在 1.4.4 节介绍了符号变量方程的解法，下面再作若干补充。

1. dsolve 解微分方程

用 dsolve 求符号变量微分方程解的基本语句格式为

s = dsolve('f1', 'f2' ⋯)

其中 f1, f2, ⋯ 代表要求解的微分方程，初始条件和自变量， s 表示所指定的输出形式。实际用法见以下的例子。在输入微分方程时，分别用 Dy, D2y, D3y 表示一阶、二阶和三阶导数。

例 1 以 t 为默认变量解微分方程

```
>> dsolve('Dy=1+y^2')
ans =
      tan(t+C1)
```

如果带有初始条件，则是

```
>> y = dsolve('Dy=1+y^2','y(0)=1')
y =
    tan(t+1/4*pi)
```

注意， y 在 MATLAB 工作内存中，而 t 不在，用指令 syms t 可以把 t 也放到工作内存中。

例 2 解有初始条件的非线性微分方程

```
>> x = dsolve('(Dx)^2+x^2=1','x(0)=0')
x =
    -sin(t)
```

```
    sin(t)
```

上面的解是一个字符串，如果想给它赋值，可以采用下面的方法。即先将它定义成符号变量的表达式，然后对符号变量赋值。

```
>> syms y
>> y=x(1)
y =
    sin(t)
>> subs(y,1:4)
ans =
      0.8415    0.9093    0.1411   -0.7568
```

例 3　对指定的自变量求解有初始条件的二阶微分方程

```
>> y = dsolve('D2y=cos(2*x)-y','y(0)=1','Dy(0)=0','x')
y =
    (1/6*cos(3*x)-1/2*cos(x))*cos(x)+(1/2*sin(x)
    +1/6*sin(3*x))*sin(x)+4/3*cos(x)
```

例 4　解三阶常微分方程

```
>> u = dsolve('D3u = u','u(0) = 1','Du(0) = -1','D2u(0) = pi','x')
u =
    1/3*pi*exp(x)-1/3*(1+pi)*3^(1/2)*exp(-1/2*x)*sin(1/2*3^(1/2)*x)
     +(1-1/3*pi)*exp(-1/2*x)*cos(1/2*3^(1/2)*x)
```

dsolve 也能解多变量的微分方程组，包括有初始条件和没有初始条件的。

例 5　解一阶线性微分方程组

```
>> s = dsolve('Df = 3*f + 4*g','Dg = -4*f + 3*g')
s =
    f: [1x1 sym]
    g: [1x1 sym]
```

计算的结果以结构数组 s 的形式返回。要想确定 f 和 g 的值可以键入

```
>> f = s.f
f =
   exp(3*t)*(cos(4*t)*C1+sin(4*t)*C2)
>> g = s.g
g =
    -exp(3*t)*(sin(4*t)*C1-cos(4*t)*C2)
```

如果需要包括初始条件的 f 和 g 的直接形式，可键入

```
>> [f,g] = dsolve('Df = 3*f + 4*g, Dg = 4*f + 3*g','f(0) = 0, g(0) = 1')
```

```
f =
    1/2*exp(7*t)-1/2*exp(-t)
g =
    1/2*exp(-t)+1/2*exp(7*t)
```

2. 调用 MAPLE 函数

扩展的符号工具箱可以直接调用 MAPLE 自身的指令。语句格式为

maple(MAPLE 指令)

查阅这些指令的分类可以键入命令

```
>> mhelp  index
```

如果要查阅每类下面的具体内容，指令是

```
>> mhelp index [ ]
```

在方括号内填入该类的名称。如果要查阅某个指令的用法，可以键入

>> mhelp + 指令名称。

下面用这个方法来画二阶厄米多项式的图形。

在 MAPLE 中，多项式都在多项式程序包中。在这里查得的结果是 index[package]，再键入

```
>> mhelp  index[package]
```

得到多项式程序包的名称是 orthopoly，要查看调用二阶厄米多项式的方法，可键入

```
>> mhelp orthopoly
```

最后可按照 MAPLE 指令的用法，键入

```
>> maple('with(orthopoly)')
ans =
      [G, H, L, P, T, U]
>> yy = maple('H(2,x)')
yy =
   4 * x^2 - 2
```

这里的 yy 就是二阶厄米多项式，如果要画它的图形，指令是

```
>> ezplot(yy)
```

如果要求 x = 1:5 时 yy 的值，做法是

```
>> subs(yy,1:5)
ans = 2    14    34    62    98
```

1.6.2 偏微分方程工具箱

偏微分方程工具箱 (PDE Toolbox) 是求解两变量偏微分方程的工具，它可以用指令进行操作，也可以在图形用户界面中进行操作，本节简单介绍在图形用户界面中的操作。

在指令窗键入 pdetool 就打开了它的指令窗如图 1.21 所示。

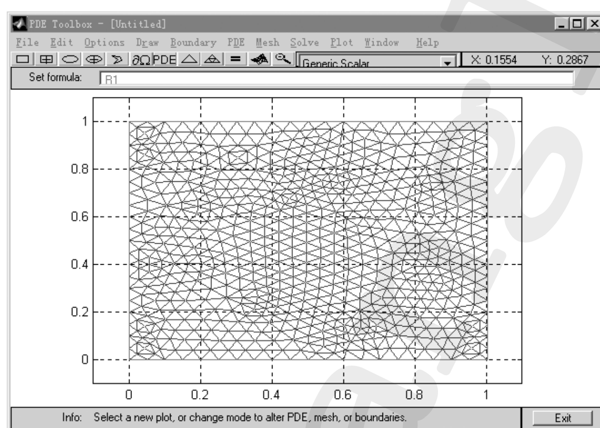


图 1.21 偏微分方程工具箱的界面

窗口中第一行是菜单，第二行是工具栏，前面是 12 个图标按钮，它们可以完成解方程的基本操作。接着的长方形框有解方程的 11 种模式，用户根据问题的类型选择，通常可用一般模式 (General Scalar)，而静电场问题则是选 electrostatics。最后是光标的坐标，供作图时选择位置用。第三行是将选定的解方程的区域用公式表示。

窗口的中间区域是用作图方式来确定方程的求解区域，x, y 轴的范围用菜单 Options 下的 Axes Limits 对话框来选择。

窗口最下面是信息栏,是对当前操作的信息提示。右下角是退出按钮 (Exit)。

解方程的操作分为以下几步：

(1) 画方程求解的区域

用按钮 1~5 来完成。画矩形用按钮 1, 2, 按钮 1 是从矩形的一个角开始画，按钮 2 是从中心开始画。比如，点击按钮 1 后，将鼠标移到指定区域，按住鼠标左键，移动光标，就会画出一个矩形，如果按住右键，就会画出一个正方形。画椭圆用按钮 3, 4, 按钮 3 是从边缘开始画，按钮 4 是从中心开始画。鼠标右键画椭圆，左键画圆。画多边形用按钮 5。点击按钮 5，将鼠标移到指定位置，按下鼠标左键，移动光标到适当的位置，然后松开左键，就会画出多边形的一条

边，再按下左键，又开始画第二条边，直至把多边形全部画好。

(2) 设定边界条件

点击第 6 个按钮，进入边界模式，然后把光标移到选定的边界上，双击鼠标左键，就打开了一个对话框，在对话框中就可以设定边界条件。如果是对所有的边界设定相同的边界条件，可以在按住 shift 键的同时再选取边界。

(3) 选定方程的类型

点击第 7 个按钮，在对话框中指定求解的方程类型。可解的方程类型有四种：椭圆型、抛物型、双曲型和本征值问题。

(4) 将网格初始化和精细化

点击第 8 个按钮，在求解区域内设置三角形网格，这是用有限元方法求解的需要，偏微分方程工具箱用的就是有限元方法。为了使解更精确，可以将网格细化，每点击一次第 9 个按钮，网格就会细化一遍，可根据需要点击一、二次。网格越细，解题的时间越长。

(5) 求解方程

点击第 10 个按钮是求解方程。默认的输出是颜色表示的二维图形。

(6) 将结果画图

点击第 11 个按钮打开的对话框可以改变结果的输出方式，其中的 Animation 是动画输出，只对含时的方程才能生效。

(7) 放大视图

第 12 个按钮可以将视图放大。

上面图标按钮的功能在菜单中都有对应的命令选项，用法相同。

下面的例子是求解二变量的热传导方程，具体的定解问题如下：

$$\begin{cases} u_{xx}+u_{yy}=0; \\ u(0,y)=0,\ u(a,y)=\mu\sin\dfrac{3\,\pi\,y}{b}; \\ u(x,0)=0,\ u(x,b)=\mu\sin\dfrac{3\,\pi\,x}{a}\cos\dfrac{\pi\,x}{a} \end{cases}$$

解题步骤如下，其中 μ, a, b 的值是解题时任意选定的：

(1) 从 Option 菜单中选择 Grid, 在作图区域画上网格，以便光标定位。再用 Axes Limits 指定 x ， y 轴的范围为 [−0.1 1.1] 。从原点开始画一个边长为 1 的正方形。

(2) 点击第 6 个按钮，进入边界模式。从菜单 Boundary 中选择菜单 Show Edge Labels ，给边界标上序号。双击边界 3 ，在对话框中选择 Dirichlet ，将边界 1 设为 sin(3*pi.*x).*cos(pi.*x), 然后关闭对话框。用同样的方法设置边界 2 的边界条件为 sin(3*pi.*y) 。

(3) 点击 PDE 按钮，在对话框中选择 Elliptic ，并将 f 设为 0，使泊松方程成为拉普拉斯方程。

(4) 点击第 8 个按钮后再点击第 9 个按钮，得到如图 1.21 所示的精细化的三角网格。

(5) 点击第 11 个按钮，选择 Hight (3_D plot) 和 Plot in x_y grid ，再点击对话框中 Plot 按钮即得到结果如图 1.22 所示。读者还可以将方程选为抛物型，并用动画来演示结果，会更为生动有趣。

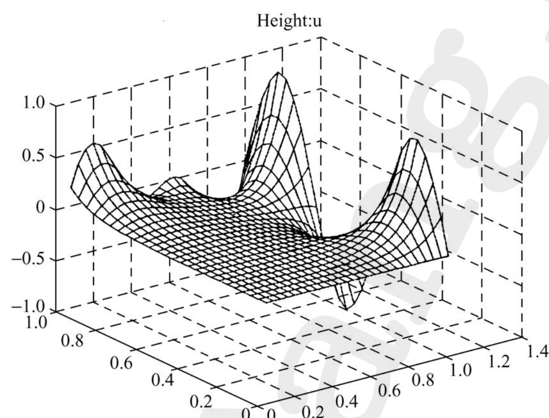


图 1.22 拉普拉斯方程的解的图形

在偏微分方程工具箱中可以用指令来完成上述任务，还可以完成更复杂的工作，限于篇幅，不再介绍。

第 2 章　模拟实验题目

2.1　阻尼斜抛运动

1. 实验题目

研究受空气阻力作用的抛体运动。设抛体质量为 m, 初速度为 $\mathbf{v}_0$ ，所受空气阻力的大小 R 与速率 v 的 n 次方成正比，即 $R=bv^n$, 其中 b 是阻尼系数。

2. 实验目的和要求

(1) 画出三种情况下即当没有空气阻力、空气阻力分别与速度一次方、二次方成正比时，抛体的运动轨迹图及水平速度随时间变化的图形。

(2) 求出三种情况的抛体轨迹的最高点，到达最高点所需的时间及所计算的抛体轨迹的终点的水平速度。

(3) 学习使用解常微分方程的指令 ode45, 尤其注意如何编写一个单独的函数文件并向函数传递参数 b,n 的值。

(4) 学习分区作图的方法，掌握在图形上加上各种标注文字的方法。

3. 解题分析

将抛体视为质点，根据牛顿运动定律，抛体的运动微分方程可统一写为

$$m\frac{\mathrm{d}^2\boldsymbol{r}}{\mathrm{d}t^2}=m\boldsymbol{g}-bv^n\frac{\mathbf{v}}{v}=m\boldsymbol{g}-bv^{n-1}\mathbf{v} \tag{2.1.1}$$

以抛出点为原点建立直角坐标系 Oxy, Ox 沿水平方向， Oy 垂直向上。于是由方程 (2.1.1) 可得两个投影方程

$$\left.\begin{aligned}\frac{\mathrm{d}^2x}{\mathrm{d}t^2}&=-\frac{b}{m}(v_x^2+v_y^2)^{\frac{n-1}{2}}v_x\\ \frac{\mathrm{d}^2y}{\mathrm{d}t^2}&=-g-\frac{b}{m}(v_x^2+v_y^2)^{\frac{n-1}{2}}v_y\end{aligned}\right\} \tag{2.1.2}$$

当 $b=0$ 即空气阻力为零时，抛体做匀变速运动，方程组 (2.1.2) 可以用代数方法求解。

当 $b\neq 0$ 时，如果 $n=1$, 则空气阻力与速率的一次方成正比 (它适用于低速情况即 v 约为 $0.1\ \mathrm{m\cdot s^{-1}}$)，方程组 (2.1.2) 可以求出解析解。做法如下：

将方程组 (2.1.2) 改写为

$$\left.\begin{aligned}\frac{\mathrm{d}^2x}{\mathrm{d}t^2}&=\frac{\mathrm{d}v_x}{\mathrm{d}t}=-\frac{b}{m}v_x\\\frac{\mathrm{d}^2y}{\mathrm{d}t^2}&=\frac{\mathrm{d}v_y}{\mathrm{d}t}=-g-\frac{b}{m}v_y\end{aligned}\right\}\tag{2.1.3}$$

取初始条件为 $t=0$ 时， $v_x=v_{x_0},v_y=v_{y_0}$, 将上式分离变量积分，得

$$\left.\begin{aligned}\frac{\mathrm{d}x}{\mathrm{d}t}&=v_x=v_{x_0}\mathrm{e}^{-\frac{b}{m}t}\\\frac{\mathrm{d}y}{\mathrm{d}t}&=v_y=\left(v_{y_0}+\frac{mg}{b}\right)\mathrm{e}^{-\frac{b}{m}t}-\frac{mg}{b}\end{aligned}\right\}\tag{2.1.4}$$

取初始条件为 $t=0$ 时， $x=y=0$, 再积分一次，得到抛体的运动学方程为

$$\left.\begin{aligned}x&=\frac{mv_{x_0}}{b}\left(1-\mathrm{e}^{-\frac{b}{m}t}\right)\\y&=\left(\frac{mv_{y_0}}{b}+\frac{m^2g}{b^2}\right)\left(1-\mathrm{e}^{-\frac{b}{m}t}\right)-\frac{mg}{b}t\end{aligned}\right\}\tag{2.1.5}$$

消去 (2.1.5) 式中的时间 t, 得到抛体的轨道方程

$$y=\left(\frac{v_{y_0}}{v_{x_0}}+\frac{mg}{bv_{x_0}}\right)x+\frac{m^2g}{b^2}\ln\left(1-\frac{bx}{mv_{x_0}}\right)\tag{2.1.6}$$

由 (2.1.4)， (2.1.5) 式可知，当 $t\to\infty$ 时， $v_x\to0$, $v_y\to-\frac{mg}{b}$，它的物理图像为：

在水平方向抛体受空气阻力的水平分力作用，水平速度不断减小而趋于零；在竖直方向抛体受重力和空气阻力的竖直分力作用，上升阶段竖直速度逐渐减小直至 为零，下降阶段的初期重力大于阻力分力使竖直速率增加。速率增加则阻力增大，直到重力与阻力大小相等且方向相反，速率达到一极限值 $v_1=\frac{mg}{b}$, 并以 v_1 匀速下降，此时轨道趋近于渐近线 $x=\frac{mv_{x_0}}{b}$ 。

由式 (2.1.4) 和式 (2.1.5) 知， $y=0$ 时的 x 值为水平射程； $v_y=\frac{\mathrm{d}y}{\mathrm{d}t}=0$ 时的 x,y 值为轨迹最高点位置；轨迹最高点位置亦可由 (2.1.6) 式，按 $\frac{\mathrm{d}y}{\mathrm{d}x}=0$ 求出。

当 $n=2$ 时，求方程组 (2.1.2) 的解析解非常困难。下面对三种情况分别计算数值解，初始条件都取为 $t=0$ 时， $x=0$, $\frac{\mathrm{d}x}{\mathrm{d}t}=3$, $y=0$, $\frac{\mathrm{d}y}{\mathrm{d}t}=5$ 。

设 $y_1=x$, $y_2=\frac{\mathrm{d}x}{\mathrm{d}t}$, $y_3=y$, $y_4=\frac{\mathrm{d}y}{\mathrm{d}t}$, 将方程组 (2.1.2) 化为 4 个一阶微分方程

$$\left.\begin{aligned}
\frac{\mathrm{d}y_1}{\mathrm{d}t} &= y_2 \\
\frac{\mathrm{d}y_2}{\mathrm{d}t} &= -\frac{b}{m}(y_2^2+y_4^2)^{\frac{n-1}{2}}y_2 \\
\frac{\mathrm{d}y_3}{\mathrm{d}t} &= y_4 \\
\frac{\mathrm{d}y_4}{\mathrm{d}t} &= -g-\frac{b}{m}(y_2^2+y_4^2)^{\frac{n-1}{2}}y_4
\end{aligned}\right\} \tag{2.1.7}$$

再将方程组 (2.1.7) 写成一个单独的函数文件，而 b 和 $p=n-1$ 作为函数文件中的参数。然后编写主程序文件 znxpyd.m 。编程的基本思路是：根据三种情况设置参数 b 和 n 的三个值，用 for 循环对三种情况重复解三遍常微分方程，解微分方程的指令是 ode45, 每次解微分方程都用题目给定的相同的初始条件，但要将不同的 b 和 n 的值传递给函数文件。解微分方程的时间范围取为 0 到 10 ，步长为 0.01 。在图形窗口用分区作图指令画了两幅图，一幅是抛体的运动轨迹；另一幅是抛体的水平速度分量随时间变化，作图用画彗星轨迹的指令 comet 。为了比较，设置了坐标轴的范围，并用 hold on 将三种运动轨迹画在一幅图内。为了便于区分三条轨迹，在图中加注了文字说明。

图 2.1 即为程序在一个图形窗口画出的两个分区图形。

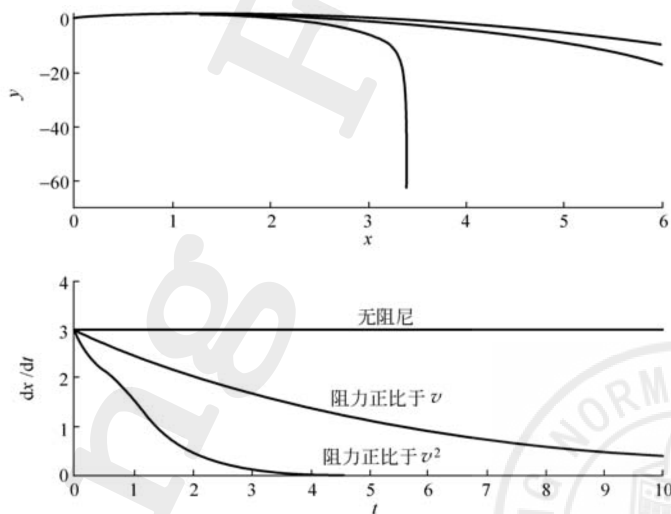


图 2.1 抛体的运动轨迹图和抛体水平速度随时间变化的图形

轨迹的最高点也就是函数文件中 y(3) 的最大值，也就是微分方程的解 y 的第三个分量 y(:, 3) ，可用指令 max 求得，每次求出的最高点的值存放在基元矩

阵 H 中。最高点对应的时间也就是上升到最高点所需的时间，求法是用指令 find 求出最高点在 y(3) 的位置即编号，再求对应这个编号的 t 值，每次求出的上升到最高点所需的时间的值存放在基元矩阵 T 。根据定义， y(2) 是水平速度，所以要求的轨迹终点的水平速度就是 y(2) 的最后一个元素 y(end, 2) 。最后在屏幕上显示的结果是

```
H =
    [1.2755]    [1.1948]    [0.9826]
T =
    [0.5100]    [0.4900]    [0.4300]
vx0 =
       [3]    [0.4060]    [6.1888e-006]
```

这表明，三种情况下的抛体的最大高度分别是 1.2755, 1.1948, 0.9826; 上升时间分别是 0.5100, 0.4900, 0.4300; 所求的轨迹终点的水平速度是 3, 0.4060, 6.1888 $\times 10^{-6}$ 。从这个结果和图形都可以看出，在给定的时间范围内，第一种情况的水平速度保持不变；第二种情况的水平速度虽然在不断地下降，但尚未达到极限值零；而第三种情况的水平速度已经基本达到极限值零。如果适当增加解微分方程的时间范围，也能使第二种情况水平速度达到极限值，这与前面解析分析的结果一致。注意由于计算的时间范围较大，抛体上升的高度较小，所以上升段的曲线不明显，如果缩短计算的时间，就可以更明显地表现上升段的曲线。

4. 思考题

(1) 将解微分方程的时间范围缩短为 0~1s ，适当改变坐标轴的设定范围，重新画图，结果有什么不同？

(2) 画出抛体的垂直速度随时间变化的图形，以观察它达到极限值的情况。

(3) 将作图指令 comet 改为 plot, 结果有什么不同？

5. 参考程序

主程序的文件名是 znxpyd.m

```
m=1;      %%小球质量
b=[0,0.2,0.2];      %%设置参数
p=[0,0,1];
px=[4.6;4.5;4.5];      %%px,py 是说明文字的坐标
py=[3.5;1.8;0.4];
strdd{1} = '无阻尼' ;      %%三组说明文字
strdd{2} = '阻力正比于 v ';
strdd{3} = '阻力正比于v^2';
```

```
figure
for i = 1 :  3      %%重复解三遍微分方程
[t,y]=ode45('znxpydfun',[0:0.01:10],[0,3,0,5],[ ],b(i),p(i),m);
H{i}=max(y(:,3))     %%求轨道的最高点
T{i}=t(find(y(:,3)==H{i}))     %%到最高点的所需时间
vx0{i}=y(end,2)     %%最终水平速度
subplot(2,1,1)     %%第一幅分区图
axis([0 6 -70 2]);     %%设置坐标轴的范围
hold on
xlabel('x')     %%标注 x,y 轴
ylabel('y')
comet(y(:,1),y(:,3));     %%画轨道的彗星轨迹
subplot(2,1,2)     %%第二幅分区图
axis([0 10 0 4])
hold on
xlabel('t')
ylabel('dx/dt')
text(px(i),py(i),strdd{i});     %%加注说明文字
comet(t,y(:,2))
end
```

函数文件是一个独立的文件，文件名为 znxpydfun.m 。

```
function    ydot=znxpydfun(t,y,flag,b,p,m)
ydot=[y(2);
      -b/m*y(2)*(y(2).^2+y(4).^2)^(p/2);
      y(4);
      -9.8-b/m*y(4)*(y(2).^2+y(4).^2)^(p/2)];
```



2.2 行星轨道

1. 实验题目

研究质点在平方反比引力场中的运动，例如行星绕太阳的运动。设质量为 m_0 的质点位于力心且固定不动，质量为 m 的质点在 m_0 产生的引力场中运动，当 m 与 m_0 相距 r 时，质点所受万有引力为 $F=\frac{Gm_0m}{r^2}$， G 为引力常量。

2. 实验目的和要求

(1) 当质点总能量大于、等于和小于零时，画出质点在平方反比引力场中的运动轨迹。

(2) 当质点总能量小于零且保持不变时，改变角动量的大小，画出质点相应的运动轨迹。

(3) 学习将极坐标变换成直角坐标的方法以及利用对称性画曲线的方法。

3. 解题分析

以力心为原点 O ，建立极坐标系，根据牛顿运动定律，可得质点的运动微分方程为

$$\left.\begin{aligned}&\frac{\mathrm{d}^2r}{\mathrm{d}t^2}-r\left(\frac{\mathrm{d}\theta}{\mathrm{d}t}\right)^2=-\frac{Gm_0}{r^2}\\&r\frac{\mathrm{d}^2\theta}{\mathrm{d}t^2}+2\frac{\mathrm{d}r}{\mathrm{d}t}\frac{\mathrm{d}\theta}{\mathrm{d}t}=0\end{aligned}\right\}\tag{2.2.1}$$

令 $y_1=r,y_2=\frac{\mathrm{d}r}{\mathrm{d}t},y_3=\theta,y_4=\frac{\mathrm{d}\theta}{\mathrm{d}t}$ ，将 (2.2.1) 式化为 4 个一阶微分方程。

$$\left.\begin{aligned}&\frac{\mathrm{d}y_1}{\mathrm{d}t}=y_2\\&\frac{\mathrm{d}y_2}{\mathrm{d}t}=y_1y_4^2-\frac{Gm_0}{y_1^2}\\&\frac{\mathrm{d}y_3}{\mathrm{d}t}=y_4\\&\frac{\mathrm{d}y_4}{\mathrm{d}t}=-\frac{2}{y_1}y_2y_4\end{aligned}\right\}\tag{2.2.2}$$

为了简单，设质点从近日点开始运动，其位置为 r_0, 速率为 v_0 ，即初始条件为 $t=0$ 时， $y_1=r_0,y_2=0,y_3=0,y_4=v_0/r_0$ 。

由近日点的位置 r_0 和速率 v_0 可得出质点的总能量

$$E=-\frac{Gm_0m}{r_0}+\frac{mv_0^2}{2}\tag{2.2.3}$$

轨道的类型由能量的符号决定：

(1) 若 $E>0$，则偏心率 $e>1$，轨道为双曲线；
(2) 若 $E=0$，则偏心率 $e=1$，轨道为抛物线；
(3) 若 $E<0$，则偏心率 $e<1$，轨道为椭圆。
对于椭圆轨道，其半长轴 a 由总能量决定

$$E=-\frac{Gm_0m}{2a} \tag{2.2.4}$$

椭圆的偏心率 e 由总能量 E 和角动量 L 共同决定

$$e=\sqrt{1+\frac{2EL^2}{(Gm_0)^2m^3}} \tag{2.2.5}$$

图 2.2 画出了总能量不同的质点运动轨迹，图 2.3 画的是总能量相同角动量不同的质点运动轨迹。

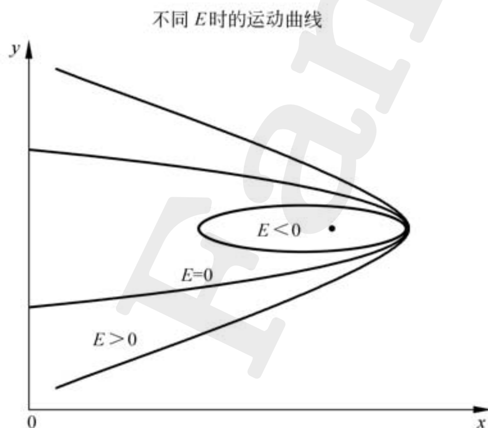


图 2.2 总能量不同的质点运动轨迹图

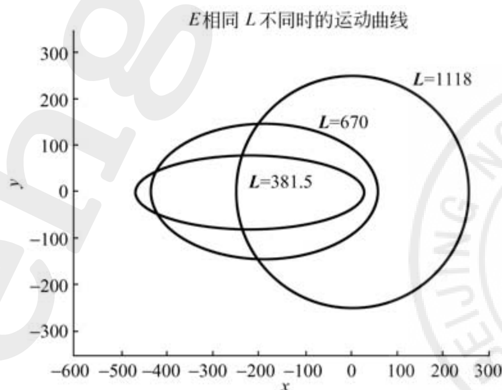


图 2.3 总能量相同角动量不同的质点运动轨迹图

不失一般性，计算时取 $m=1, G=1, m_0=5000$ 。计算程序分为两段，第一段计算 $E=-18,0,22$ 时质点的运动轨迹。近日点的位置取为 $r_0=100$ ，利用方程 (2.2.3) 求出质点在近日点的速率 v_0，根据这些初始条件可解出方程 (2.2.2) 并画图。由于质点是从近日点开始运动，所以只计算了一半轨迹，而另一半轨迹是利用对称性用指令 flipud 将数据颠倒顺序而得到的。

第二段程序是 $E=-10$ 时，不同的 L 所对应的轨迹。由于能量不变，所有的轨道具有相同的半长轴 a，可以用式 (2.2.4) 算出 $a=250$ 。从式 (2.2.5) 可以看出，由于 E 为负值，角动量 L 越大，离心率 e 越小。当 $r_0=a$ 时， $e=0$, $L=1118$ ，对应的轨道是圆；当 $r_0<a$ 时， $0<e<1$，轨道是椭圆。分别取 $r_0=50$ ， 15 ，利用 (2.2.3) 式求出相应的速率为

$$v_0=\sqrt{2E+\frac{2Gm_0}{r_0}}$$

代入数值后得 $v_0=13.4,\ 25.4$ ，相应的角动量为 $L=mr_0v_0=670,\ 381.5$ 。利用这些数据解微分方程并画图，可以验证 e 和 L 的关系。

4. 思考题

取总能量 $E=10$，画出角动量 L 不同的质点的运动轨迹。

5. 参考程序

主程序的文件名是 xxgd.m 。

```
G=1;M=5000;r0=100;
E=[-18,0,22];
v0=sqrt(2*E+2*G*M/r0);      %%不同的 E 所对应的速率
figure
axis([-400,200,-1000,1000]);
xlabel('x');
ylabel('y');
title(' 不同 E 时的不同曲线 ');
s{1}='E<0';s{2}='E=0'; s{3}='E>0';
hold on
    %%标出太阳位置
sun=line(0,0,'color','r','marker','.','markersize',20);
for i=1:3
    [t,u]=ode45('xxgdfun',[0:0.05:120],[r0,0,0,v0(i)/r0],[ ],G,M);
    [x,y]=pol2cart(u(:,3),u(:,1));      %%将极坐标变换为直角坐标
```

```
    X=[flipud(x);x];     %%用对称性求出另一半的轨道的数据
    Y=[-flipud(y);y];
    text(-100-(i-1)*100,0-(i-1)*300,s{i});
    comet(X,Y);
    plot(X,Y)
end
E=-10;
a=-G*M/2/E;     %%计算椭圆的半长轴
r0=[a,50,15];
v0=sqrt(2*E + 2*G*M./r0);     %%轨道为椭圆时对应的速率
figure
axis([-600,300,-300,300]);
xlabel('x');
ylabel('y');
title('E 相同 L 不同时的不同曲线 ');
hold on
axis equal
sun=line(0,0,'color','r','marker','.','markersize',20);
for i=1:3
    [t,u]=ode45('xxgdfun',[0:0.05:180],...
      [r0(i),0,0,v0(i)/r0(i)],[ ],G,M);
    [x,y]=pol2cart(u(:,3),u(:,1));
    X=[flipud(x);x];
    Y=[-flipud(y);y];
    plot(X,Y)
end
```

函数文件是一个独立的文件，文件名为 xxgdfun.m 。

```
function ydot=xxgdfun(t,y,flag,G,M)
ydot=[y(2);
     y(1)*y(4)^2-G*M/(y(1)^2);
     y(4);
     -2*y(2)*y(4)/y(1)];
```



2.3 粒子散射

1. 实验题目

研究平方反比斥力场中粒子的运动。以 α 粒子在重核场中的运动为例，设重核位于力心且固定不动， α 粒子的质量为 m，它到重核的距离为 r，所受到库仑斥力为 $F=\frac{k}{r^2}$， k 为由库仑定律确定的常量。

2. 实验目的和要求

(1) 画出 α 粒子在不同初始条件下的轨道，通过改变初始条件来研究影响散射角的因素。

(2) 学习根据解决问题的需要来选择坐标系，本题就是选择直角坐标系而不是极坐标系。

3. 解题分析

根据牛顿运动定律， α 粒子矢量形式的运动微分方程为

$$\frac{\mathrm{d}^2\boldsymbol{r}}{\mathrm{d}t^2}=\frac{k}{m}\frac{\boldsymbol{r}}{r^3} \tag{2.3.1}$$

为便于求解，建立直角坐标系，原点 O 位于力心重核处，则 (2.3.1) 式在直角坐标系 Oxy 中的投影方程为

$$\left.\begin{aligned}\frac{\mathrm{d}^2x}{\mathrm{d}t^2}&=\frac{k}{m}\frac{x}{\left(x^2+y^2\right)^{3/2}}\\ \frac{\mathrm{d}^2y}{\mathrm{d}t^2}&=\frac{k}{m}\frac{y}{\left(x^2+y^2\right)^{3/2}}\end{aligned}\right\} \tag{2.3.2}$$

令 $y_1=x$, $y_2=\frac{\mathrm{d}x}{\mathrm{d}t}$, $y_3=y$, $y_4=\frac{\mathrm{d}y}{\mathrm{d}t}$，则 (2.3.2) 式可写成

$$\left.\begin{aligned}\frac{\mathrm{d}y_1}{\mathrm{d}t}&=y_2\\ \frac{\mathrm{d}y_2}{\mathrm{d}t}&=\frac{k}{m}\frac{y_1}{\left(y_1^2+y_3^2\right)^{3/2}}\\ \frac{\mathrm{d}y_3}{\mathrm{d}t}&=y_4\\ \frac{\mathrm{d}y_4}{\mathrm{d}t}&=\frac{k}{m}\frac{y_3}{\left(y_1^2+y_3^2\right)^{3/2}}\end{aligned}\right\} \tag{2.3.3}$$

可令 α 粒子沿 Ox 方向入射，入射速率为 v_0，则初始条件为 $y_1=x_0$, $y_2=v_0$, $y_3=y_0$, $y_4=0$。实际程序十分简单，为了能得到多条粒子的运动轨迹，程序中给出了多个初始条件。程序运行的结果如图 2.4 所示。

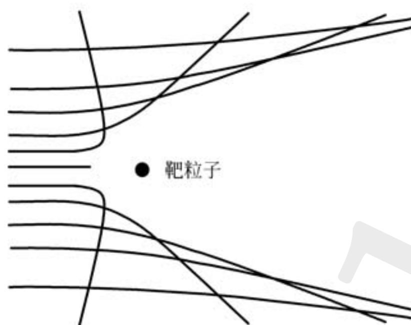


图 2.4 α 粒子的散射轨道

实际研究 α 粒子散射时，α 粒子都是从重核力场之外入射，即 $|x_0|$ 的值必须足够大，使得初始时刻的轨道近乎于平行的直线，α 粒子离开力场后的轨道也为直线，此直线与 x 轴的夹角即出射方向相对入射方向的偏转角，也就是散射角。改变 y_0, v_0, k, m, 可以定性观察它们对散射角的影响。

4. 思考题

如果将方程 (2.3.3) 改成 2.2 节 (2.2.2) 式的形式，能不能求得结果？

5. 参考程序

主程序的文件名是 alzss.m 。

```
function alzss
figure
y0=[-10,1,10,0;        %%不同的初始位置
-10,1,2,0;
-10,1,0,0;
-10,1,-10,0;
-10,1,-4,0;
-10,1,4,0;
-10,1,15,0;
-10,1,-15,0;
-10,1,-2,0;
-10,1,7,0;
-10,1,-7,0];
line(0,0,'marker','.','markersize',50,'color','r');
text(2,0,' 靶粒子 ','fontsize',14 );
```

```
xlabel('x');ylabel('y');
hold on
axis([-10 20 -20 20])
for i=1:11
   [t,y]=ode23('alzssfun',[0:.1:32],y0(i,:),[ ],3);
   plot(y(:,1),y(:,3))
end
```

函数文件是一个独立的文件，文件名为 alzssfun.m, 其中 $p = k/m$ 。

```
function ydot=alzssfun(t,y,flag,p)
ydot=[y(2);
     p*y(1)/sqrt(y(1).*y(1)+y(3).*y(3)).^3;
     y(4);
     p*y(3)/sqrt(y(1).*y(1)+y(3).*y(3)).^3];
```




2.4 水星近日点的进动

1. 实验题目

研究水星近日点的进动。由于广义相对论对万有引力定律的修正，引起水星运动轨道的进动，水星的空间轨道不再是闭合的椭圆轨道。广义相对论对万有引力的修正可以归结为在原来的运动方程中增加一个小的修正项 ε/r^4，其中 $\varepsilon=3Gm_0mh^2/c^2$ 是小量， G 为万有引力常量， m_0 为太阳质量， m 为水星质量， c 为真空中的光速， h 为水星掠面速度的两倍。水星运动轨道的进动如图 2.5 所示。

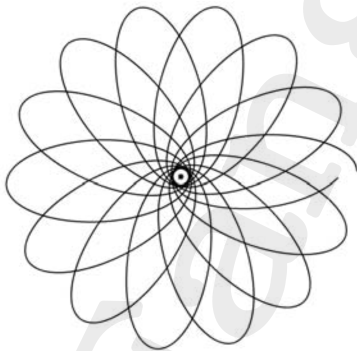


图 2.5 水星运动的轨道

2. 实验目的和要求

画出水星运动轨道。验证只要质点在有心力场中所受的力与平方反比引力有微小偏离，其轨道就不是闭合的椭圆，从而证明广义相对论对万有引力定律的修正将引起椭圆轨道的进动。

3. 解题分析

利用比尼公式求得水星的极坐标方程为

$$\frac{\mathrm{d}^2u}{\mathrm{d}\theta^2}+u=\frac{Gm_0}{h^2}+\frac{3Gm_0}{c^2}u^2$$

引入

$$a=\frac{Gm_0}{h^2},\ b=\frac{3Gm_0}{c^2}$$

则轨道方程可写为

$$\frac{\mathrm{d}^2u}{\mathrm{d}\theta^2}=-u+a+bu^2$$

等价于两个一阶方程，设 $u_1 = u, u_2 = \frac{\mathrm{d}u}{\mathrm{d}\theta}$，则上式为

$$\left.\begin{aligned}\frac{\mathrm{d}u_1}{\mathrm{d}\theta} &= u_2 \\ \frac{\mathrm{d}u_2}{\mathrm{d}\theta} &= -u_1 + a + b{u_1}^2\end{aligned}\right\}$$

数值求解时，由于实际修正项非常之小，需适当选取参数，夸张地展示轨道的进动情况。

4. 思考题

(1) 本题能不能用极坐标的数值直接作图？为什么？

(2) 改变方程中 b 值的大小，运动轨迹将如何变化？

5. 参考程序

主程序的文件名是 sxjd.m 。

```
[theta,u]=ode45('sxjdfun',[0:pi/100:30*pi],[0.1,0]);
figure
axis equal
axis off
hold on
plot(0,0,'*r')       %%太阳位置
[x,y]=pol2cart(theta,1./u(:,1));       %%极坐标转换为直角坐标
plot(x,y)
```

函数文件是一个独立的文件，文件名为 sxjdfun.m 。

```
function ydot=sxjdnfun(theta,u)
a=1;
b=0.06;
ydot=[u(2);
     -u(1)+a+b*u(1)^2];
```




2.5　霍曼轨道

1. 实验题目

把一艘飞船从地球送到金星，最节省能量的方法不是让飞船沿直线飞向该行星，而是让它航行于半椭圆轨道。该轨道与地球轨道和金星轨道相切，这种半椭圆双切轨道称为霍曼轨道。霍曼轨道是一种最省能量的行星际间的输运轨道，可以从外行星向内行星输运，也可以从内行星向外行星输运。

如图 2.6 所示，内、外两圆是金星和地球运行的轨道，它们的半径分别为 0.72 AU 和 1.00 AU(1AU= 日地距离)，它们的运行方向由图中箭头表示。从日心坐标系看，静止在地球上的飞船先是以地球的轨道速度 v_E 作圆周运动。霍曼轨道就是以 EV 为长轴的椭圆轨道，它与两行星轨道相切。飞船在 E 点进行调速，进入霍曼轨道。利用开普勒第三定律计算出从 E 点运行到 V 点所需的时间。选择适当的发射时刻，使飞船运行到 V 点时正好与金星相遇，在 V 点再次进行调速，使飞船与金星一起沿金星轨道运动。

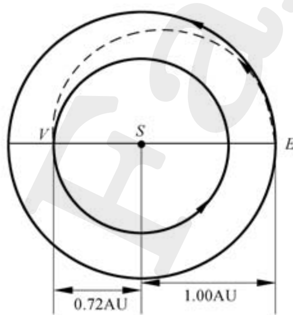


图 2.6　霍曼轨道的示意图

2. 实验目的和要求

模拟从地球发射飞船沿霍曼轨道到达金星并与之同步运动。

3. 解题分析

将地球和金星绕太阳的运动近似看作匀速圆周运动。飞船脱离地球后的运动近似看作只受太阳的引力作用。

地球的运动：$r_1 = 1\ \text{AU} = 1.496 \times 10^8\ \text{km}$，$\omega_1 = \dfrac{2\pi}{T_1}$，$T_1 = 365.26\ \text{d}$

金星的运动：$r_2 = 0.72\ \text{AU}$，$\omega_2 = \dfrac{2\pi}{T_2}$，$T_2 = 224.7\ \text{d}$

选用极坐标系，飞船的运动微分方程为

$$\left.\begin{aligned}&\frac{\mathrm{d}^2r}{\mathrm{d}t^2}-r\left(\frac{\mathrm{d}\theta}{\mathrm{d}t}\right)^2=-\frac{Gm_0}{r^2}\\&r\frac{\mathrm{d}^2\theta}{\mathrm{d}t^2}+2\frac{\mathrm{d}r}{\mathrm{d}t}\frac{\mathrm{d}\theta}{\mathrm{d}t}=0\end{aligned}\right\}$$

式中 G 为万有引力常数， m_0 为太阳质量。作数值计算时还需将方程化为一阶方程，即实验 2.2 中列出的方程组 (2.2.2) 式。

下面来计算实现这一霍曼轨道所需的两次调速。在 E 点飞船已具有地球的轨道速率 v_E ，其较准确的数值为 29.6 km/s ，飞船欲沿长轴等于 EV 的椭圆运动需要的总能量为

$$E=-\frac{Gm_0m}{2a}=-\frac{Gm_0m}{1.72r_1}$$

从而可得出飞船在 E 点所需的速度 v_1

$$\frac{1}{2}mv_1^2-\frac{Gm_0m}{r_1}=-\frac{Gm_0m}{1.72r_1}$$

$$v_1^2=\left(2-\frac{1}{0.86}\right)\frac{Gm_0}{r_1}$$

$$v_1\approx 0.92v_E\approx 27.2\ \mathrm{km/s}$$

因此所需调速为

$$\Delta v_1=27.2-29.6=-2.4\ \mathrm{km/s}$$

即需沿地球运动的相反方向以 2.4 km/s 的速率发射飞船。飞船沿椭圆轨道飞行到达 V 点的速率 v_2 可从以下计算得出

$$\frac{1}{2}mv_2^2-\frac{Gm_0m}{0.72r_1}=-\frac{Gm_0m}{1.72r_1}$$

$$v_2^2=\left(\frac{1}{0.36}-\frac{1}{0.86}\right)\frac{Gm_0}{r_1}=1.62v_E^2$$

$$v_2\approx 1.27v_E\approx 37.7\ \mathrm{km/s}$$

而金星的轨道速率为

$$v_V=\sqrt{\frac{Gm_0}{0.72r_1}}=34.9\ \mathrm{km/s}$$

因此需要进行第二次调速，调速量为

$$\Delta v_2=34.9-37.7=-2.8\ \mathrm{km/s}$$

即飞船必须启动制动火箭，使其速率减少 2.8 km/s 。

计算程序用三次循环来计算飞船的三种发射速度所对应的轨道，计算出的轨道画在图 2.7 中。在图 2.7 中，每一幅图都有相应的文字说明。为了体现立体的效果，用指令 view 来改变观察角度。从图上可以看出，若在 E 点飞船的发射

速度小于 v_1，则飞船的轨道不能到达 V 点，所以不会与金星轨道相切；若在 E 点飞船的发射速度大于 v_1，则飞船的轨道会超过 V 点，也不能与金星轨道相切；只有在 E 点飞船的发射速度等于 v_1，飞船的轨道才能到达 V 点与金星轨道相切。

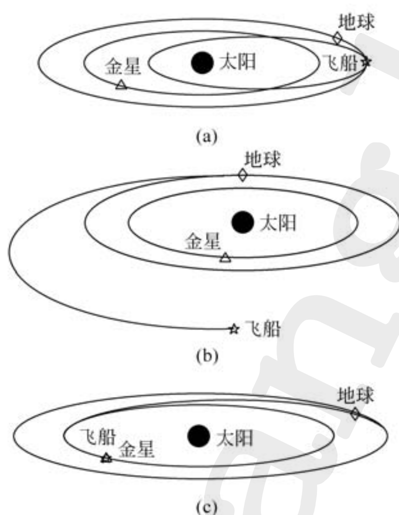


图 2.7 飞船的轨道

(a) 飞船发射速度小于 v_1 时的轨道；(b) 飞船发射速度大于 v_1 时的轨道；(c) 飞船发射速度等于 v_1 时的轨道

计算中所采用的数据如下：

解方程传递的参数为 $G = G\,m_0$, 取值为 8900。

由开普勒第三定律，$T = 2\pi\sqrt{a^3/GM}$, 可求出飞船在霍曼轨道上运行的时间为 $t3 = T/2 \approx 0.84$。在程序中时间的步长为 0.01, 故得时间的步数为 84，程序作动画时将它调整为 86。此后飞船将脱离霍曼轨道而沿金星的轨道运行。地球公转的周期为 $t1 \approx 2.3$。地球和金星的初始角速度分别为 2.98 rad/s 和 4.88 rad/s，地球与金星的初始位置的极角分别取为 0 和 $(0.01 - 1/3)\,\pi$。飞船初始的角速度随需要而选取。这些数据根据实际情况计算得出，但略有改动。

4. *思考题*

将图形恢复为二维的形状，所得的结果是什么样？

5. *参考程序*

主程序的文件名是 hmgd.m。

```
figure
for j=1:3      %%循环三次计算不同的飞船发射速度所对应的轨道
   time1=[2.3;2.1;2.3];     %%地球与金星三次运动的时间
   time2=[2.3;2.1;0.84];     %%飞船三次运动的时间
   v=[2.1;3.5;2.74];     %%飞船的发射速度

    %%下面依次求地球、金星与飞船的轨道
   [t1,z1]=ode45('hmgdfun',[0:0.01:time1(j)],[10,0,0,2.98],[ ],8900);
   [t2,z2]=ode45('hmgdfun',[0:0.01:time1(j)],[7.2,0,...
        (pi*(0.01-1/3)),4.88],[ ],8900);
   [t3,z3]=ode45('hmgdfun',[0:0.01:time2(j)],[10,0,0,v(j)],[ ],8900);
   [x1,u1]=pol2cart(z1(:,3),z1(:,1));     %%将极坐标变为直角坐标
   [x2,u2]=pol2cart(z2(:,3),z2(:,1));
   [x3,u3]=pol2cart(z3(:,3),z3(:,1));
   plot(x1(:),u1(:),'y',x2(:),u2(:),'y',x3(:),u3(:),'b')     %%画轨迹
   hold on
   axis off

    %%以下是太阳，地球，金星，飞船的图形句柄
   sun=line(0,0,'color','r','erasemode','xor','marker',...
         '.','markersize',80);
   earth=line(10,0,'color','b','marker','.','erasemode',...
         'xor','markersize',40);
   venus=line(7.2,0,'color','m','marker','.','erasemode',...
         'xor','markersize',30);
   ship=line(10,0,'color','r','marker','p','erasemode',...
         'xor','markersize',10);

    %%以下 text 中的内容均为图示标注
   if j==1
      view(0,58)
      title('霍曼轨道')
      text(-10,-10,-1,'方位角： 0 度；仰角： 58 度')
      text(-4,19,1,'依次为太阳，地球，金星，飞船')
      text(-8,32,1,'当飞船速度小于相切速度时，...
          宇宙飞船轨道可以与金星')
```

```
    text(-10,28,1,' 轨道相交。虽然这时宇宙飞船...
        可能与金星相遇，但飞船 ')
    text(-10,24,1,' 要降在金星上，它的速度需要...
        改变很大，需要极大能量.  ')
    sun00=line(-8,35,'color','r','marker','.','markersize',18);
    earth11=line(-7,35,'color','b','marker','.','markersize',14);
    venus22=line(-6,35,'color','m','marker','.','markersize',12);
    ship33=line(-5,35,'color','r','marker','p','markersize',10);
elseif j==2
    view(-90,50)
    text(-65,16,1,' 方位角：  -90 度；仰角：  50 度 ');
    text(-4,8,1,' 依次为太阳，地球，金星，飞船 ');
    text(-1,15,1,' 当飞船速度大于相切速度时,...
        飞船的轨道将不与金星轨道相交。 ')
    sun00=line(-4,12,1,'color','r','marker','.','markersize',18);
    earth11=line(-4,11,1,'color','b','marker','.','markersize',14);
    venus22=line(-4,10,1,'color','m','marker','.','markersize',12);
    ship33=line(-4,9,1,'color','r','marker','p','markersize',10);
else j==3
    view(1,54)
    text(-10,-10,-1,' 方位角：  1 度；仰角：  54 度 ')
    text(-6,12,1,' 依次为太阳，地球，金星，飞船 ')
    text(-10,16,1,' 当飞船速度等于相切速度时,...
        飞船的轨道将与金星轨道相切。 ')
    sun00=line(-10,27,'color','r','marker','.','markersize',18);
    earth11=line(-9,27,'color','b','marker','.','markersize',14);
    venus22=line(-8,27,'color','m','marker','.','markersize',12);
    ship33=line(-7,27,'color','r','marker','p','markersize',10);
end

n1=length(t3);
for i=1:n1       %%模拟地球、金星与飞船的运动
    set(earth,'xdata',x1(i),'ydata',u1(i));
    set(venus,'xdata',x2(i),'ydata',u2(i));
    set(ship,'xdata',x3(i),'ydata',u3(i));
    drawnow;
```

```
    pause(0.1);
  end

 %%在第三种情形(霍曼轨道),飞船与金星相遇后将按金星的轨道运动
  if j==3
    n2=length(t2);
    for i=86:n2
      set(earth,'xdata',x1(i),'ydata',u1(i));
      set(venus,'xdata',x2(i),'ydata',u2(i));
      set(ship,'xdata',x2(i),'ydata',u2(i));
      drawnow;
      pause(0.1);
    end
  end
  cla
end
```

函数文件是一个独立的文件，文件名为 hmgdfun.m 。

```
function rdot= hmgdfun(t,r,flag,G)
rdot=[r(2);
      r(1)*r(4)*r(4)-G/r(1)/r(1);
      r(4);
      -2*r(2)*r(4)/r(1)];
```




2.6 行星引力加速

1. 实验题目

研究飞船在飞临某个行星附近时所产生的行星“引力加速”现象。设地球、木星、土星、天王星、海王星等行星都在同一平面内围绕太阳做匀速圆周运动，飞船除受太阳引力作用外，原则上还受这些行星引力的作用，实际上只有当飞船飞临某行星附近时，该行星的引力才足以吸引飞船，使之加速。因此需要等待各大行星位置排列成有利阵形并合理地设计飞船的航行轨道，使之能途经某行星，经加速后又能到达目标行星。图 2.8 是飞船被途经行星“引力加速”的轨迹。

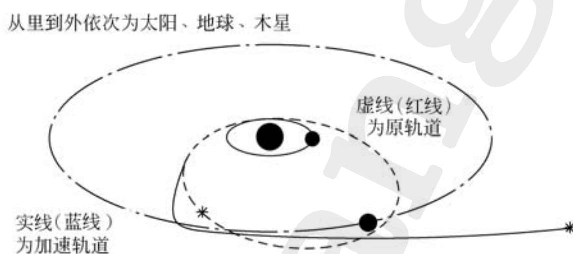


图 2.8 飞船被途经行星“引力加速”的轨迹

2. 实验目的和要求

演示飞船被途经行星“引力加速”后，轨道发生改变的现象。若加速后飞船的总能量从小于零变为大于零，则飞船轨道将从椭圆改变为双曲线，从而大大缩短到达外行星的航行时间。

3. 解题分析

现说明引力加速原理。当飞船沿椭圆轨道飞行与途经某行星相遇时（设该行星为木星），在进入木星引力为主的范围后，飞船被木星短时俘获，这时，飞船一面被木星携带着以巨大的木星轨道速度运动，一面被木星较强的引力改变着运动方向。在被俘获的较短时间内，太阳对飞船的引力作用相对木星引力作用很小。而由于飞船质量相对于木星质量很小，飞船的运动几乎不改变木星的运动，在短时间内木星可视为作匀速直线运动，木星坐标系（即原点建立在木星中心的平动坐标系）可视为惯性系。飞船相对该惯性系的运动与带电粒子在核场中的散射类似，遵守机械能守恒定律。因此飞船再次飞离木星引力场时的相对速度的大小 U_{f}，与它进入木星引力场的相对速度大小 U_{i} 是相等的，木星引力场的作用仅使相对速度的方向发生偏转。

如图 2.9 所示，设木星的轨道速度为 $\boldsymbol{V}_\mathrm{j}$，飞船与木星相互作用前后相对日心系的速度分别为 $\boldsymbol{V}_\mathrm{i}$ 和 $\boldsymbol{U}_\mathrm{f}$，则

$$\boldsymbol{V}_\mathrm{i} = \boldsymbol{V}_\mathrm{j} + \boldsymbol{U}_\mathrm{i}$$

$$\boldsymbol{V}_\mathrm{f} = \boldsymbol{V}_\mathrm{j} + \boldsymbol{U}_\mathrm{f}$$

飞船获得的动能增量为

$$\Delta T = m\boldsymbol{V}_\mathrm{j}\cdot(\boldsymbol{U}_\mathrm{f} - \boldsymbol{U}_\mathrm{i})$$

通过改变瞄准距离，速度大小的增量可接近 V_i，经木星后飞船的轨道变为双曲线，从而大大缩短飞行时间。

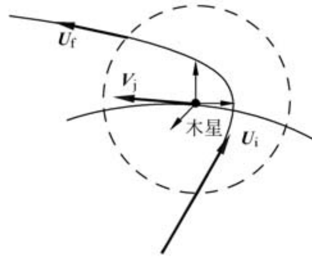


图 2.9　飞船经过行星附近时速度发生改变

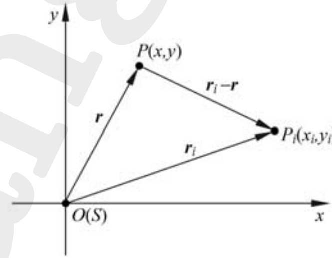


图 2.10　飞船与各行星相对位置的示意图

在用数值计算的方法求飞船的运动时，可认为飞船在太阳、地球、木星、土星、天王星、海王星等多个星体的引力作用下运动。由于这些星体均按已知规律运动，所以用数值计算的方法求解飞船的运动在原则上是可行的。以太阳为原点，建立平面直角坐标系（惯性系），设某星体 P_i 的质量为 m_i，其位置用矢量 $\boldsymbol{r}_i(x_i, y_i)$ 表示。飞船 P 的位置用矢量 $\boldsymbol{r}(x,y)$ 表示。如图 2.10 所示。于是描述飞船运动的矢量形式运动微分方程为

$$\frac{\mathrm{d}^2\boldsymbol{r}}{\mathrm{d}t^2} = \sum_i \frac{Gm_i\,(\boldsymbol{r}_i - \boldsymbol{r})}{|\boldsymbol{r}_i - \boldsymbol{r}|^3} \tag{2.6.1}$$

其投影方程为

$$\left.\begin{aligned}\frac{\mathrm{d}^2x}{\mathrm{d}t^2} &= \sum_i \frac{Gm_i\,(x_i - x)}{\left[(x_i - x)^2 + (y_i - y)^2\right]^{3/2}}\\ \frac{\mathrm{d}^2y}{\mathrm{d}t^2} &= \sum_i \frac{Gm_i\,(y_i - y)}{\left[(x_i - x)^2 + (y_i - y)^2\right]^{3/2}}\end{aligned}\right\} \tag{2.6.2}$$

设第 i 个星体的轨道半径为 r_i，其匀速圆周运动的角速度为 ω_i，初位置相

对于 Ox 轴的极角为 θ_{i0}，则上式中的

$$x_i = r_i \cos(\omega_i t + \theta_{i0}), \quad y_i = r_i \sin(\omega_i t + \theta_{i0})$$

都为已知。

设 $y_1 = x$, $y_2 = \frac{\mathrm{d}x}{\mathrm{d}t}$, $y_3 = y$, $y_4 = \frac{\mathrm{d}y}{\mathrm{d}t}$，将 (2.6.2) 式化为

$$\left.\begin{aligned}
&\frac{\mathrm{d}y_1}{\mathrm{d}t} = y_2 \\
&\frac{\mathrm{d}y_2}{\mathrm{d}t} = \sum_i \frac{Gm_i(x_i - y_1)}{\left[(x_i - y_1)^2 + (y_i - y_3)^2\right]^{3/2}} \\
&\frac{\mathrm{d}y_3}{\mathrm{d}t} = y_4 \\
&\frac{\mathrm{d}y_4}{\mathrm{d}t} = \sum_i \frac{Gm_i(y_i - y_3)}{\left[(x_i - y_1)^2 + (y_i - y_3)^2\right]^{3/2}}
\end{aligned}\right\}$$

如果考虑木星对飞船的引力加速，实际计算只需考虑太阳和木星两个星体引力的作用，其他星体的作用可忽略不计。程序的编写比较简单，只要正确地设置各种参数即可。为了比较，程序中还计算了没有经过行星“引力加速”的飞船轨道，方法是在原来的计算中取木星的质量为零即可。图 2.8 就是程序运行的结果。

4. 思考题

本节用于计算行星引力加速的函数文件与 2.2 节行星轨道中用于计算的函数文件有何不同？

5. 参考程序

主程序的文件名是 xxyljs.m 。

```
                  %%program
G=6.67e-11;       %%万有引力常数
Msun=1.99e30;     %%太阳质量
Mear=5.976e24;      %%地球质量
Mjup=6e29;        %%放大的木星质量
unit=1.4959787e11;       %%日地距离
rear=unit;
rjup=5.203*unit;       %%木星与太阳距离
tear0=3.1;        %%地球初始角坐标
tjup0=-1.06;        %%木星初始角坐标
```

```
figure;     %%画各行星的轨道
ti=0:pi/60:2*pi;
hold on
axis([-1e12,1e12,-1e12,1e12]);
axis off
view([86,48])
title(' 放大的加速轨迹 ','fontsize',14);
text(-5.43e+11,-8.89e+11,0.5,' 从里到外依次为太阳，地球，木星 ');
    %%画地球与木星轨道
plot(rear*cos(ti),rear*sin(ti),'c',rjup*cos(ti),rjup*sin(ti),'c');

xo=(rear+10.37e6)*cos(tear0);     %%初始时刻飞船 x 坐标
xd=3.7e9*(-sin(tear0));     %%初始时刻飞船 x 方向速度
yo=(rear+10.37e6)*sin(tear0);     %%初始时刻飞船 y 坐标
yd=3.7e9*(cos(tear0));     %%初始时刻飞船 y 方向速度
[t,x]=ode23('xxjsfun2',[0:1:1000],[xo,xd,yo,yd],...
[ ],tear0,tjup0,Msun,Mear,Mjup,rear,rjup,G);   %%求飞船的运动轨迹

    %%模拟地球、木星与飞船的运动
sun = line(0,0,'color','r','marker','.',...
   'markersize',45,'erasemode','xor');
Earth = line(rear*cos(tear0), rear*sin(tear0), 'color', 'b',...
   'marker', '.', 'markersize', 25,'erasemode','xor');
Jupiter = line(rjup*cos(tjup0), rjup*sin(tjup0), 'color', 'g',...
   'marker', '.', 'markersize',30,'erasemode','xor');
voyagera = line(rear*cos(tear0), rear*sin(tear0), 'color', 'k',...
   'marker', '*', 'markersize',7,'erasemode','xor');
voyageraa = line(rear*cos(tear0), rear*sin(tear0), 'color', 'b',...
   'marker', '.', 'markersize',1,'erasemode','none');
text(8e+11,-10e+11,' 蓝线为加速轨道 ');

for t=1:1001
   thetaear=tear0+t/365*2*pi;
   thetajup=tjup0+t/(11.86*365)*2*pi;
   set(Earth,'xdata',rear*cos(thetaear),'ydata',rear*sin(thetaear));
   set(Jupiter,'xdata',rjup*cos(thetajup),'ydata',rjup*sin(thetajup));
```

```
   set(voyagera,'xdata',x(t,1),'ydata',x(t,3));
   set(voyageraa,'xdata',x(t,1),'ydata',x(t,3));
   drawnow;
end
plot(x(:,1),x(:,3),'b')

Mjup=0;      %%令木星质量为零，再计算飞船轨道
[t,x]=ode23('xxjsfun2',[0:1:3000],[xo,xd,yo,yd],...
   [ ],tear0,tjup0,Msun,Mear,Mjup,rear,rjup,G);
plot(x(:,1),x(:,3),'r')
text(-1e+11,3e+11,' 红线为原轨道 ')
```

函数文件是一个独立的文件，文件名为 xxjsfun2.m 。

```
function xdot=xxjsfun2(t,x,flag,tear0,tjup0,Msun,...
Mear,Mjup,rear,rjup,G)
xdot=[x(2);
      3600*24*3600*24*(G*Msun*(-x(1))/(x(1)^2+x(3)^2)^1.5...
       +G*Mear*(rear*cos(tear0+t/365*2*pi)-...
       x(1))/((rear*cos(tear0+t/365*2*pi)-x(1))^2...
       +(rear*sin(tear0+t/365*2*pi)-x(3))^2)^1.5...
       +G*Mjup*(rjup*cos(tjup0+t/(11.86*365)*2*pi)-...
       x(1))/((rjup*cos(tjup0+t/(11.86*365)*2*pi)-...
       x(1))^2+(rjup*sin(tjup0+t/(11.86*365)*2*pi)...
       -x(3))^2)^1.5);
      x(4);
      3600*24*3600*24*(G*Msun*(-x(3))/(x(1)^2+x(3)^2)^1.5...
       +G*Mear*(rear*sin(tear0+t/365*2*pi)-...
       x(3))/((rear*cos(tear0+t/365*2*pi)-x(1))^2...
       +(rear*sin(tear0+t/365*2*pi)-x(3))^2)^1.5...
       +G*Mjup*(rjup*sin(tjup0+t/(11.86*365)*2*pi)-...
       x(3))/((rjup*cos(tjup0+t/(11.86*365)*2*pi)-...
       x(1))^2+(rjup*sin(tjup0+t/(11.86*365)*2*pi)...
       -x(3))^2)^1.5)];
```

2.7 带电粒子在电磁场中的运动

1. 实验题目

研究带电粒子在均匀稳定电磁场中的运动。设带电粒子质量为 m，带电量为 q。电磁场的电场强度为 $\boldsymbol{E}$，磁感应强度 $\boldsymbol{B}$。分三种情况考虑：

(1) 电场强度和磁感应强度都不为零；

(2) 电场强度为零，磁感应强度不为零；

(3) 电场强度不为零，磁感应强度为零。

2. 实验目的和要求

(1) 研究带电粒子在均匀稳定电磁场中的运动情况，画出粒子运动轨迹。

(2) 学习用指令 axes 在图形窗口的不同位置设定坐标轴来画多个图形。

3. 解题分析

在电磁场中带电粒子的运动微分方程为

$$m\frac{\mathrm{d}^2\boldsymbol{r}}{\mathrm{d}t^2} = q\boldsymbol{E} + q\boldsymbol{v}\times\boldsymbol{B} \tag{2.7.1}$$

以场中某点为原点，以 $\boldsymbol{E}$ 为 Oy 方向，$\boldsymbol{B}$ 为 Oz 方向建立坐标系 $Oxyz$，令 $\omega = qB/m$，则 (2.7.1) 式的投影方程为

$$\left.\begin{aligned} &\frac{\mathrm{d}^2x}{\mathrm{d}t^2} = \omega\frac{\mathrm{d}y}{\mathrm{d}t} \\ &\frac{\mathrm{d}^2y}{\mathrm{d}t^2} = \frac{qE}{m} - \omega\frac{\mathrm{d}x}{\mathrm{d}t} \\ &\frac{\mathrm{d}^2z}{\mathrm{d}t^2} = 0 \end{aligned}\right\} \tag{2.7.2}$$

(2.7.2) 式为线性微分方程组，可求其解析解。将 (2.7.2) 式中第一式积分求出 $\frac{\mathrm{d}x}{\mathrm{d}t} = \omega y + C_1$，代入第二式得

$$\frac{\mathrm{d}^2y}{\mathrm{d}t^2} + \omega^2 y = \frac{qE}{m} - \omega C_1 = \omega\left(\frac{E}{B} - C_1\right)$$

其解为相应齐次方程通解加上该非齐次方程特解

$$y = C_2\cos(\omega t + C_3) + \frac{1}{\omega}\left(\frac{E}{B} - C_1\right) \tag{2.7.3}$$

把上式代入 $\frac{\mathrm{d}x}{\mathrm{d}t} = \omega y + C_1$，积分后可求出

$$x = C_2\sin(\omega t + C_3) + \frac{E}{B}t + C_4 \tag{2.7.4}$$

由 (2.7.2) 式中第三式积分可求出

$$z = C_5 t + C_6 \tag{2.7.5}$$

(2.7.3) 式 ~ (2.7.5) 式为粒子的运动学方程，积分常数 $C_1 \sim C_6$ 由初始条件定出。

下面对粒子的运动图像加以分析。粒子运动微分方程 (2.7.1) 式有一特解 $\boldsymbol{v}_d$，它是由 $\boldsymbol{E} + \boldsymbol{v} \times \boldsymbol{B} = 0$ 所决定的常量，$\boldsymbol{v}_d = \dfrac{E}{B}\boldsymbol{i}$（$\boldsymbol{E} \times \boldsymbol{B}$ 方向）。由粒子运动学方程可知，粒子的运动为沿 $\boldsymbol{B}$ 方向的匀速直线运动、绕磁感应线的匀速圆周运动和沿 $\boldsymbol{E} \times \boldsymbol{B}$ 方向的漂移运动的叠加。

令 $y_1 = x, y_2 = \mathrm{d}x/\mathrm{d}t, y_3 = y, y_4 = \mathrm{d}y/\mathrm{d}t, y_5 = z, y_6 = \mathrm{d}z/\mathrm{d}t$，则 (2.7.2) 式成为：

$$\left.\begin{aligned}
\frac{\mathrm{d}y_1}{\mathrm{d}t} &= y_2 \\
\frac{\mathrm{d}y_2}{\mathrm{d}t} &= \omega y_4 \\
\frac{\mathrm{d}y_3}{\mathrm{d}t} &= y_4 \\
\frac{\mathrm{d}y_4}{\mathrm{d}t} &= \frac{qE}{m} - \omega y_2 \\
\frac{\mathrm{d}y_5}{\mathrm{d}t} &= y_6 \\
\frac{\mathrm{d}y_6}{\mathrm{d}t} &= 0
\end{aligned}\right\}$$

求解时按题目要求分三种情况进行讨论：

（1）$\boldsymbol{E} \neq 0, \boldsymbol{B} \neq 0$

（2）$\boldsymbol{E} = 0, \boldsymbol{B} \neq 0$

（3）$\boldsymbol{E} \neq 0, \boldsymbol{B} = 0$

并把数值计算结果与解析结果进行对照。本题程序没有新的技巧，程序运行结果如图 2.11 所示。

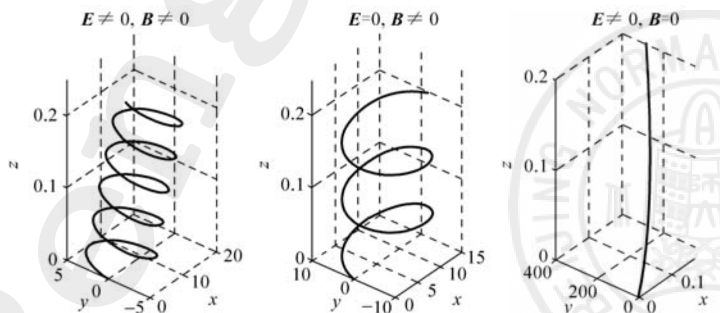


图 2.11　带电粒子在电磁场中的运动

4. 思考题

对比指令 axes 与 subplot 的用法。

5. 参考程序

主程序的文件名是 dcc.m 。

```
%%program
q=1.6e-2; m=0.02;
B=[2;1;0]; E=[1;0;1];
figure
strd{1} = 'E \neq 0, B \neq 0';
strd{2} = 'E=0, B \neq 0';
strd{3} = 'E \neq 0, B=0';
for i=1:3
    [t,y]=ode23('dccfun',[0:0.1:20],[0,0.01,0,6,0,0.01],...
         [],q,m,B(i),E(i));
         %%设置轴的范围
    axes('unit', 'normalized', 'position', [ 0.0293+(i-1)*...
    0.3240 0.062 0.2786 0.6583 ]);
    plot3(y(:,1),y(:,3),y(:,5),'linewidth',2);
    grid on
    title(strd{i},'fontsize',12,'fontweight','demi');
    view([-51,18]);
end
```

函数文件是一个独立的文件，文件名为 dccfun.m 。

```
function ydot=dccfun(t,y,flag,q,m,b,e)
ydot=[y(2);
     q*b*y(4)/m;
     y(4);
     q*e/m-q*b*y(2)/m;
     y(6);
     0];
```



2.8 落体偏东

1. 实验题目

研究落体偏东现象。地球上物体由于受地球自转的影响而不再沿铅垂（竖直）方向下落，质点落地位置较垂足偏东。试在地球上纬度为 λ 处研究这一现象，让物体从高度为 h 处的 P 点自由下落。

2. 实验目的和要求

(1) 研究科氏力对质点自由下落运动的影响，计算并图示落体的偏东效应。

(2) 研究落体偏东效应与纬度的关系。

3. 解题分析

在地球上建立坐标系 $Oxyz$：原点 O 点在地面， Ox 轴指南， Oy 轴指东，Oz 轴垂直地面向上。由于地球是非惯性系，质点除受重力外还受惯性力作用。当把 $m\boldsymbol{g}$ 理解为表观重力时，惯性力中只需考虑科氏力的影响。故质点相对地球的运动微分方程为

$$m\frac{\mathrm{d}^2\boldsymbol{r}}{\mathrm{d}t^2}=m\boldsymbol{g}-2m\boldsymbol{\omega}\times\boldsymbol{v} \tag{2.8.1}$$

其中 $\boldsymbol{\omega}$ 为地球自转角速度。于是可得 (2.8.1) 式在 $Oxyz$ 系内的投影方程为

$$\left.\begin{aligned} m\frac{\mathrm{d}^2x}{\mathrm{d}t^2}&=2m\omega\frac{\mathrm{d}y}{\mathrm{d}t}\sin\lambda\\ m\frac{\mathrm{d}^2y}{\mathrm{d}t^2}&=-2m\omega(\frac{\mathrm{d}z}{\mathrm{d}t}\cos\lambda+\frac{\mathrm{d}x}{\mathrm{d}t}\sin\lambda)\\ m\frac{\mathrm{d}^2z}{\mathrm{d}t^2}&=2m\omega\frac{\mathrm{d}y}{\mathrm{d}t}\cos\lambda-mg \end{aligned}\right\} \tag{2.8.2}$$

(2.8.2) 式是线性微分方程组，可求解析解。设初始条件为 $t=0$ 时 $x=y=0$, $z=h$, $\frac{\mathrm{d}x}{\mathrm{d}t}=\frac{\mathrm{d}y}{\mathrm{d}t}=\frac{\mathrm{d}z}{\mathrm{d}t}=0$ 。将 (2.8.2) 中三个式子积分一次并定出积分常数，得

$$\left.\begin{aligned} \frac{\mathrm{d}x}{\mathrm{d}t}&=2\omega y\sin\lambda\\ \frac{\mathrm{d}y}{\mathrm{d}t}&=-2\omega\left[(z-h)\cos\lambda+x\sin\lambda\right]\\ \frac{\mathrm{d}z}{\mathrm{d}t}&=2\omega y\cos\lambda-gt \end{aligned}\right\} \tag{2.8.3}$$

将上式代入 (2.8.2) 式得

$$\left.\begin{aligned}\frac{\mathrm{d}^2x}{\mathrm{d}t^2}&=-4\omega^2\left[(z-h)\cos\lambda+x\sin\lambda\right]\sin\lambda\\\frac{\mathrm{d}^2y}{\mathrm{d}t^2}&=2gt\omega\cos\lambda-4\omega^2y\\\frac{\mathrm{d}^2z}{\mathrm{d}t^2}&=-g-4\omega^2\left[(z-h)\cos\lambda+x\sin\lambda\right]\cos\lambda\end{aligned}\right\}$$

由于 $\omega=7.29\times10^{-5}$， ω^2 项可以忽略，于是上式化为

$$\left.\begin{aligned}\frac{\mathrm{d}^2x}{\mathrm{d}t^2}&=0\\\frac{\mathrm{d}^2y}{\mathrm{d}t^2}&=2gt\omega\cos\lambda\\\frac{\mathrm{d}^2z}{\mathrm{d}t^2}&=-g\end{aligned}\right\}$$

对上式积分两次并定出积分常数，得到

$$\left.\begin{aligned}x&=0\\y&=\frac{1}{3}gt^3\omega\cos\lambda\\z&=h-\frac{1}{2}gt^2\end{aligned}\right\}$$

在忽略 ω^2 项的条件下，科氏力对 Ox 和 Oz 方向的运动几乎没有影响， y 值反映出其偏东效应。由上式消去时间 t 可得质点运动的轨道方程。

$$\left.\begin{aligned}x&=0\\y^2&=\frac{8}{9}\frac{\omega^2\cos^2\lambda}{g}(h-z)^3\end{aligned}\right\}$$

下面对 (2.8.2) 式求数值解并作图可得到直观的图像。将 (2.8.2) 式化为

$$\left.\begin{aligned}\frac{\mathrm{d}y_1}{\mathrm{d}t}&=y_2\\\frac{\mathrm{d}y_2}{\mathrm{d}t}&=2\omega y_4\sin\lambda\\\frac{\mathrm{d}y_3}{\mathrm{d}t}&=y_4\\\frac{\mathrm{d}y_4}{\mathrm{d}t}&=-2\omega\left(y_6\cos\lambda+y_2\sin\lambda\right)\\\frac{\mathrm{d}y_5}{\mathrm{d}t}&=y_6\\\frac{\mathrm{d}y_6}{\mathrm{d}t}&=2\omega y_4\cos\lambda-g\end{aligned}\right\}$$

在前述初始条件下求解即可。程序的编写比较简单，没有新的要求。

由于数值求解并未做忽略 ω^2 项的近似，所以数值解不但可求出落体的偏东效应，还包括偏南等高阶效应，作图时可以把偏南效应也反映出来。但读者应注意，偏南效应并没有实际价值。因为我们在建立质点的动力学方程时并未考虑月球对落体的影响、重力加速度 g 的不均匀性等的作用，所以求解运动微分方程时得出的与 ω^2 或更高阶项相关的效应实际上可以不予考虑。

图 2.12 是程序运行的结果，图中的两条曲线在计算机屏幕上是彩色的。

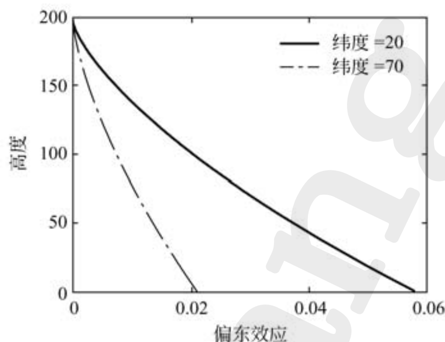


图 2.12 落体偏东效应

4. 思考题

如果地球的自转角速度取得过大，可能会出现与落体偏东效应不同的结果，试分析一下。

5. 参考程序

主程序的文件名是 ltpd.m 。

```
h=200;
w=2*pi/(60*60*24);
g=9.8;
tt=sqrt(2*h/g);
t=0:0.01:tt;
r0=[0,0,0,0,h,0];
wd1=20*2*pi/360;
wd2=70*2*pi/360;
[t,r1]=ode23('ltpdfun',t,r0,[ ],wd1,w);
[t,r2]=ode23('ltpdfun',t,r0,[ ],wd2,w);
```

```
plot(r1(:,3),r1(:,5),'r',r2(:,3),r2(:,5));
xlabel('偏东效应');          %%x 轴上的标注
ylabel('高度');
legend('纬度 =20','纬度 =70');      %%图例标注
```

函数文件是一个独立的文件，文件名为 ltpdfun.m 。

```
function y=ltpdfun(t,r,flag,a,w)
g=9.8;
ydot=[y(2);
     2*w*y(4)*sin(r);
     y(4);
     -2*w*(y(6)*cos(r)+y(2)*sin(r));
     y(6);
     2*w*y(4)*cos(r)-g];
```




2.9 小环在转动大环上的运动

1. 实验题目

如图 2.13 所示，质量为 m 的小环，套在半径为 R 的光滑大圆环上，并可沿大环滑动，大环在水平面内以匀角速度 $\boldsymbol{\omega}$ 绕环上 O 点转动。小环相对大环的位置可用过小环的大圆环半径与过 O 点的大环直径间的夹角 θ 表示。

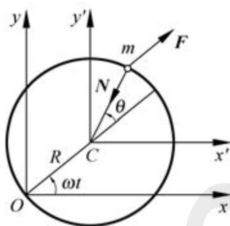


图 2.13 在大环上建立两个坐标系

2. 实验目的和要求

(1) 求出小环的运动方程并模拟小环沿大圆环运动。

(2) 注意在模拟动画中正确地表示静止坐标系与运动坐标系的关系。

(3) 学习从三维的角度观察二维的图形或动画。

3. 解题分析

以大环上的固定点 O 为原点，建立与地面固连的直角坐标系 Oxy，以大环中心 C 为原点建立平动坐标系 $Cx'y'$，如图 2.13 所示。在非惯性系 $Cx'y'$ 内，在水平面中小环受到大环约束力 $\boldsymbol{N}$ 和惯性力 $\boldsymbol{F}$。惯性力的大小为 $F = m\omega^2 R$，沿 OC 方向。故在非惯性系 $Cx'y'$ 内小环的运动微分方程为

$$\frac{\mathrm{d}^2\theta}{\mathrm{d}t^2} = -\omega^2 \sin\theta$$

令 $y_1 = \theta, y_2 = \dfrac{\mathrm{d}\theta}{\mathrm{d}t}$，则上式化为

$$\left.\begin{aligned}\frac{\mathrm{d}y_1}{\mathrm{d}t} &= y_2\\ \frac{\mathrm{d}y_2}{\mathrm{d}t} &= -\omega^2 \sin y_1\end{aligned}\right\}$$

这是一个简单的常微分方程，很容易求出数值解。在推导方程的过程中，没有作小角度近似，因此也能应用于小环偏离平衡位置较大的情形。如果要得

出小环作微振动的情况，小环偏离平衡位置不宜太远。小环的初速度以及大环的转速可以通过试验适当选取。需要注意的是如何正确地作出动画模拟。为了表示出小环在运动中对平衡位置的偏离，程序中用一条过 O ， C 两点的直径来表示运动中平衡位置的变化。大环是用参数方程画出，所用参数是大环的圆心角 ϕ ，半径为 $R=2$, 初始时刻大环的圆心在点 $(R,0)$ ，所以大环上各点的坐标为 $(R+R\cos\phi, R\sin\phi)$, 运动中大环的圆心在点 $(R\cos\omega t, R\sin\omega t)$ ，所以大环上各点的坐标为 $(R\cos\omega t+R\cos\phi, R\sin\omega t+R\sin\phi)$, 小环的位置可以用类似的方法计算，但是要同时考虑小环自身的运动及大环的转动所带来的影响。为了达到更好的观察效果，对二维的图形用指令 view 进行三维观察。图 2.14 是程序运行结果。

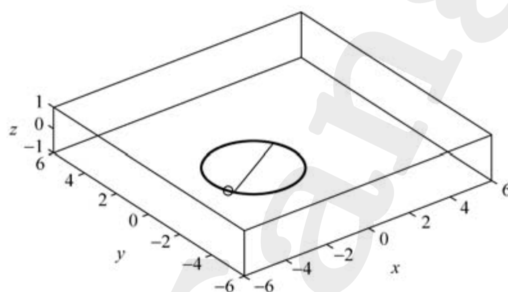


图 2.14　小环与大环的模拟运动

4. 思考题

(1) 改变大环的转动速度，会对小环的运动产生什么影响？

(2) 逐渐增加小环的初始速度，会对小环的运动产生什么影响？

(3) 如果小环的初始位置偏离平衡位置较远，小环的运动是什么样？

5. 参考程序

主程序的文件名是 dhyxh.m 。

```
R=2;      %%大环半径
theta=pi/6 ;  w=0.1;       %%小环的初始位置与初始角速度
[t,y]=ode23('dhyxhfun',[0:0.2:100],[theta,w],[],pi/3);
figure
axis([-6 6 -6 6,-1,1 ])
hold on
axis equal;
```

```
box on
view(3);      %%三维视角
phi=0:0.1:2*pi;      %%大环的圆心角
xh=R*cos(phi);     yh=R*sin(phi);      %%大环的参数方程
    %%大环, 小环及直径 OC 的初始位置
DY=line(R+xh,yh,'linestyle','-','color','b','linewidth',3,...
    'erasemode','xor');
SY=line(R+R*cos(theta),R*sin(theta),'color','r','marker','o',...
    'markersize',10,'linewidth',2,'erasemode','xor');
L=line([0,2*R],[0,0],'color','k','erasemode','xor');
for i=1:2:500      %%实时动画
   set(DY,'Xdata',xh+R*cos(w*t(i)),'Ydata',yh+R*sin(w*t(i)));
   set(SY,'Xdata',R*cos(y(i,1)+w*t(i))+R*cos(w*t(i)),...
      'Ydata',R*sin(y(i,1)+w*t(i))+R*sin(w*t(i)));
   set(L,'Xdata',[0,2*R*cos(w*t(i))],'Ydata',[0,2*R*sin(w*t(i))]);
   drawnow;
   pause(0.01)      %%暂停以减慢动画的放映速度
end
```

函数文件是一个独立的文件，文件名为 dhyxhfun.m 。

```
function  ydot=dhyxhfun(t,y,flag,omega)
ydot=[y(2);
      -omega^2*sin(y(1))];
```

2.10 小球在弹簧顶端木块上的弹性跳动

1. 实验题目

如图 2.15 所示，地面上有垂直放置的一弹簧，弹簧上端固定有一个木块。有一个小球从空中自由下落，与木块发生弹性碰撞。设小球及木块均只沿垂直方向运动，木块保持平动且顶面与地面平行，研究小球与木块运动。小球质量为 m_1 ，木块质量为 m_2 。弹簧劲度系数为 k 。

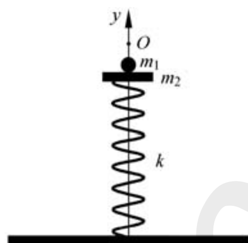


图 2.15 小球与弹簧上的木块的运动

2. 实验目的和要求

(1) 画出小球与木块的位移曲线并模拟两物体的运动状况。

(2) 学习用指令 events 来控制求解微分方程的进程。

3. 解题分析

以弹簧自然伸长位置的上端作为原点， y 轴竖直向上，设置坐标系如图 2.15 所示。以 y_1, y_2 分别表示小球与木块的位置，根据牛顿定律，可列出除碰撞瞬时之外小球与木块的运动微分方程

$$\left.\begin{aligned} m_1\frac{\mathrm{d}^2y_1}{\mathrm{d}t^2} &= -m_1g \\ m_2\frac{\mathrm{d}^2y_2}{\mathrm{d}t^2} &= -m_2g - ky_2 \end{aligned}\right\} \tag{2.10.1}$$

令 $y_3 = \mathrm{d}y_1/\mathrm{d}t,\ y_4 = \mathrm{d}y_2/\mathrm{d}t$ ，则运动微分方程表示为

$$\left.\begin{aligned} \frac{\mathrm{d}y_1}{\mathrm{d}t} &= y_3 \\ \frac{\mathrm{d}y_2}{\mathrm{d}t} &= y_4 \\ \frac{\mathrm{d}y_3}{\mathrm{d}t} &= -9.8 \\ \frac{\mathrm{d}y_4}{\mathrm{d}t} &= -9.8 - \frac{k}{m_2}y_2 \end{aligned}\right\} \tag{2.10.2}$$

小球与木块相碰条件为 $y_1 = y_2$ 。在弹性碰撞过程中动量守恒、机械能守恒。又由于碰撞过程十分短暂，所以可以忽略碰撞过程中二物体高度的变化，则在碰撞过程中有

$$\left.\begin{aligned} &m_1 v_{1y} + m_2 v_{2y} = m_1 v_{1y}^{'} + m_2 v_{2y}^{'} \\ &\frac{1}{2}m_1 v_{1y}^2 + \frac{1}{2}m_2 v_{2y}^2 = \frac{1}{2}m_1 v_{1y}^{'2} + \frac{1}{2}m_2 v_{2y}^{'2} \end{aligned}\right\} \tag{2.10.3}$$

令 $y_3 = v_{1y} = \mathrm{d}y_1/\mathrm{d}t$ ， $y_4 = v_{2y} = \mathrm{d}y_2/\mathrm{d}t$ ，得

$$\left.\begin{aligned} &m_1 y_3 + m_2 y_4 = m_1 y_3^{'} + m_2 y_4^{'} \\ &m_1 y_3^2 + m_2 y_4^2 = m_1 y_3^{'2} + m_2 y_4^{'2} \end{aligned}\right\} \tag{2.10.4}$$

在编写程序时，首先给定小球和木块开始运动的初始条件，由运动微分方程组求出它们的解。把相碰条件 $y_1 = y_2$ 作为指令 events 控制求解进程的条件。当满足条件 $y_1 = y_2$ 时，求解方程的过程停止。按照弹性碰撞的条件求出这时两物体碰撞后的速度 $v_{1y}^{'}$ 和 $v_{2y}^{'}$ 。然后用此刻的位置 $y_1^{'} = y_2^{'} = y_1 = y_2$ 和速度 $v_{1y}^{'}$ 和 $v_{2y}^{'}$ 作为下一次解方程的初始条件。如此继续反复进行即得出小球与木块的运动方程。程序在画图时对不同的曲线用了不同的颜色，指令 legend 会在显示的图例中将曲线颜色的对应关系表示出来，由于本书的图形是黑白的，所以在图 2.16 中不能看出这种对应关系。只有计算机的屏幕上才能看出曲线的颜色及其对应关系。程序运行的结果如图 2.15 和图 2.16 所示。

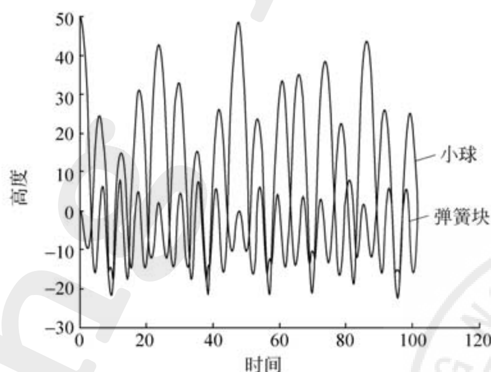


图 2.16　小球与木块的位移图像

4. 思考题

(1) 指令 events 的功能是什么，如果不使用它，在解微分方程的过程中会遇到什么问题？

(2) 将本题改为自由下落的小球与作简谐振动的桌面作弹性碰撞，试研究它们的运动。

5. 参考程序

主程序的文件名是 xqythk.m 。

```
h0=50; m1=20;      %%小球的高度和质量
k=60; m2=50;       %%弹簧倔强系数和木块质量
tstart=0;  tfinal=1000;      %%解微分方程的起止时间
y0=[h0;0;0;0];       %%存放初始条件的变量
tout=tstart;       %%存放时间序列的变量
yout=y0.';      %%存放小球和木块的位移及速度序列的变量
options=odeset('Events','on');       %%开启事件判断功能
for i=1:25
    [t,y,event]=ode45('xqythkfun',[tstart:0.03:tfinal],y0,options);
    tout=[tout;t(2:end)];       %%将每次得到的数据依次存在同一矩阵
    yout=[yout;y(2:end,:)];
    y0(1)=y(end,1);     y0(2)=y(end,2);     %%下一次弹跳的初位移
    v10=y(end,3);     v20=y(end,4);
         %%由动量守恒与机械能守恒解出下一次弹跳的初速度
    y0(3)= (-m2*v10+2*m2*v20+m1*v10)/(m2+m1);
    y0(4)=(2*m1*v10+m2*v20-v20*m1)/(m2+m1);
    tstart=t(end);
end
figure
ylabel(' 高度 ');    xlabel(' 时间 ');
hold on
plot(tout,yout(:,1),tout,yout(:,2));      %%画小球与木块的位移曲线
legend(' 小球 ',' 弹簧块 ');      %%在图形上显示图例
figure      %%运动模拟
axis([-1 1 -50 h0+10]);    axis off
hold on

    %%下面的三句是用正弦函数画弹簧的初位置
yt1=-45:0.3:0;    xt1=0.06*sin(yt1);
tanhuang=line(xt1,yt1,'color','k','erasemode','xor','linewidth',2);
qiu=line(0,yout(1,1)+4,'color','b','erasemode','xor',...
```

```
'marker','.','markersize',50);      %%小球的初位置
tank = line( [ -0.1,0.1 ], [ yout(1,2), yout(1,2) ], 'color',...
[ 0.3 0.1 0.5],'erasemode', 'xor','linewidth',8);      %%木块的初位置
ground=line([-0.5,.5],[-50,-50],'color',[0.6 0.1 0.2],...
'linewidth',20);      %%画地面

for i=1:length(tout)      %%实时动画
    yt=-45:0.3:yout(i,2);       %%运动中弹簧的位置
    xt=0.06*sin((yt-yout(i,2))*(-45)./(-45-yout(i,2)));
    set(tanhuang,'xdata',xt,'ydata',yt);      %%画运动中弹簧
    set(qiu,'ydata',yout(i,1)+4);      %%画运动中小球
    set(tank,'ydata',[yout(i,2),yout(i,2)]);      %%画运动中木块
    drawnow;
end
```

函数文件是一个独立的文件，文件名为 xqythkfun.m 。文件的格式采用了 odefile 模板的格式，用指令 switch 来执行何时运用事件判断的功能。

```
function vararout=xqythkfun(t,y,flag)
switch flag
case ''
    varargout1=f(t,y);
case 'events'
    [varargout1:3]=events(t,y);
otherwise
    error(['Unknown flag ' '' flag '' '.']);
end

function ydot=f(t,y)      %%计算微分方程的子函数
k=100;  m1=30;  m2=50;
ydot=[y(3);  y(4);  -9.8;  -9.8-(k/m2)*y(2);];

    %%事件判断子函数
function [value,isterminal,direction]=events(t,y)
Q=y(1)-y(2);    value=Q;       %%当 Q 为 0 时，解微分方程终止
isterminal=1;      %%开启判断终止功能
direction=-1;      %%由 Q 减小的方向终止
```

2.11 大摆角单摆

1. 实验题目

研究大摆角单摆的运动。设单摆由质量为 m 的摆锤和长为 l 的轻杆构成。

2. 实验目的和要求

(1) 研究单摆总能量对运动的影响。

(2) 研究单摆的摆角对运动周期的影响。

(3) 学习利用总能量等于势能与动能之和的关系来画相图。

3. 解题分析

以悬点为原点 O 建立极坐标系，极轴 Ox 竖直向下，以 θ 表示单摆偏离平衡位置的角度。单摆的运动微分方程为

$$\frac{\mathrm{d}^2\theta}{\mathrm{d}t^2}+\frac{g}{l}\sin\theta=0 \tag{2.11.1}$$

其中 g 为重力加速度。

单摆的势能为 $V=mgl\,(1-\cos\theta)$，在这个余弦势场中，单摆的总能量 E 决定了单摆的三种运动状态。下面在 θ 与 $\frac{\mathrm{d}\theta}{\mathrm{d}t}$ 构成的相平面分三种情况讨论单摆的运动。

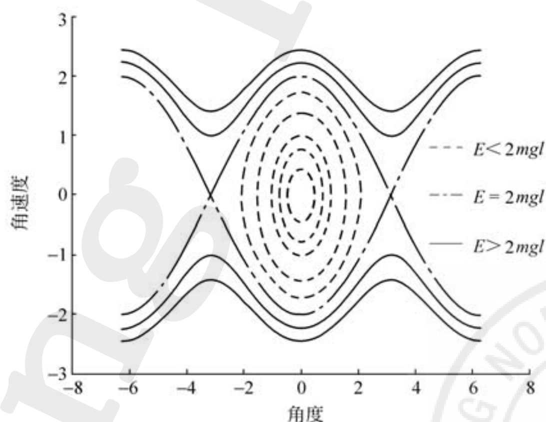


图 2.17 总能量不同的单摆所对应的相图

(1) $E<2mgl$，摆锤在 $-\pi\sim\pi$ 的势阱中作周期运动，其相轨迹为一闭合曲线，能量较小（对应小摆角运动）时为椭圆。

(2) $E>2mgl$，摆锤在势场中作定向运动，且 θ 可以趋向 $\pm\infty$。其相轨迹是两条不相交的曲线，对应两个不同的运动方向。$\mathrm{d}\theta/\mathrm{d}t>0$ 表示向 θ 的正方向运动；$\mathrm{d}\theta/\mathrm{d}t<0$ 表示向 θ 的负方向运动。

(3) $E=2mgl$，摆锤运动处于临界状态。当其 θ 值对应势能曲线的极大值时，摆锤速度为零。下一时刻的运动具有不确定性，摆锤即可能沿原运动方向运动，也可能改变运动方向。这种情况下摆锤的相轨迹为两条对 $\mathrm{d}\theta/\mathrm{d}t=0$ 直线对称的、在 $(\theta=n\pi,\mathrm{d}\theta/\mathrm{d}t=0)$ 点相交的曲线。

这些结果就是图 2.17 中的曲线。

当单摆处于第一种运动状态时，其运动周期与最大摆角有关。经数值计算可验证：若摆角不大于 5°，在小摆角 $(\theta\ll 1)$ 近似下，$\sin\theta\approx\theta$，运动可视为简谐振动，其固有圆频率为 $\omega=\sqrt{g/l}$。若摆角增大，则运动周期变长。其关系如图 2.18。

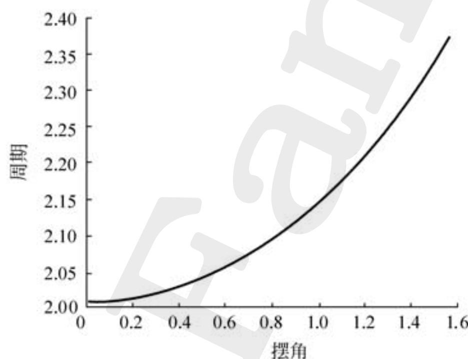


图 2.18 单摆的周期与最大摆角的关系

下面求 (2.11.1) 式数值解，令 $y_1=\theta, y_2=\dfrac{\mathrm{d}\theta}{\mathrm{d}t}$，则 (2.11.1) 式成为

$$\left.\begin{aligned}\frac{\mathrm{d}y_1}{\mathrm{d}t}&=y_2\\ \frac{\mathrm{d}y_2}{\mathrm{d}t}&=-\frac{g}{l}\sin y_1\end{aligned}\right\}\tag{2.11.2}$$

计算程序分为三部分，第一部分画图 2.17 中的相图。所用的数据不是解微分方程的结果，而是利用总能量等于动能与势能之和得到

$$E=\frac{1}{2m}\left(ml\frac{\mathrm{d}\theta}{\mathrm{d}t}\right)^2+mgl(1-\cos\theta)$$

$$\frac{\mathrm{d}\theta}{\mathrm{d}t}=\pm\sqrt{\frac{2g}{l}\left(\frac{E}{mgl}-1+\cos\theta\right)}$$

令 $e=\dfrac{E}{mgl}$, 则除了相差一个常数因子之外，可以认为

$$\frac{\mathrm{d}\theta}{\mathrm{d}t}=\pm\sqrt{e-1+\cos\theta} \tag{2.11.3}$$

前面讨论的总能量的三种情况分别对应 $e<2$, $e>2$, $e=2$ 。每给定一个 e 值，就可以画出一条曲线。

程序的第二部分是画单摆在不同的初始位移下所对应的位移曲线 (见图 2.19)，可以看出，摆角越大则周期越长。改变初始条件就能得到不同的结果。

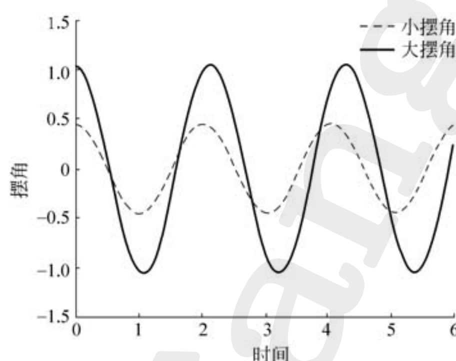


图 2.19 最大摆角不同的位移曲线

程序的第三部分是画图 2.18 中最大摆角与周期的关系。这里仍然使用了指令 events，当单摆从正 θ 最大值的一端运动到另一端 (θ 为负，但绝对值最大) 过程中，速度始终为负，并且从零逐渐减小到负的最大值然后又增加到零，这段时间正好是半个周期，所以程序中用速度作为被 events 判断的事件变量。当它为零时，停止解微分方程，即 isterminal=1；而 direction=1 表示速度从负值增加到零。

4. 思考题

(1) 计算最大摆角与周期的关系能否找到其它方法？请试一试。

(2) 能否利用微分方程的解来画相图，请试一试。

5. 参考程序

主程序的文件名是 djddb.m 。

```
    %%第一部分程序是画相图
figure
axis([-8 8 -2 2])
```

```
hold on
    %%文字标注，说明不同颜色的曲线所对应的系统总能量
plot([4.5,5.2],[0.8,0.8],'g',[4.5,5.2],[0,0],'r',...
   [4.5,5.2],[-0.8 -0.8],'b');
text(5.3,0.8,'E<2mgl');
text(5.3,0,'E=2mgl');
text(5.3,-0.8,'E>2mgl');
xlabel(' θ ');
ylabel('d θ /dt');
ydot=inline('sqrt(abs(2*(E-1+cos(x))))','x','E');   %%公式 (2.11.3)
e=[3, 2.5, 2, 1.5, 1, 0.5, 0.3, 0.1];     %%取不同能量
for k=1:8
   if k>3      %%对应 E<2mgl
       Q{k}=acos(1-e(k));     %%动能为零的点
       X=linspace(-Q{k},Q{k},300);      %%确定 θ 角的变化范围
       y=ydot(X,e(k));      %%相图中曲线的纵坐标数据
       plot(X,y,'g',X,-y,'g')      %%利用对称性画曲线
    elseif k==3      %%对应 E=2mgl
       X=linspace(-2*pi,2*pi,300);
       y=ydot(X,e(k));
       plot(X,y,'r',X,-y,'r')
    else      %%对应 E>2mgl
       X=linspace(-2*pi,2*pi,300);
       y=ydot(X,e(k));
       plot(X,y,'b',X,-y,'b')
    end
end

    %%程序第二部分解不同初始角度下的微分方程
[t1,w1]=ode45('djddbfun',[0:0.001:6],[pi/7,0],[ ]);
[t2,w2]=ode45('djddbfun',[0:0.001:6],[pi/3,0],[ ]);
figure      %%画不同角度下的位移曲线
plot(t1,w1(:,1),t2,w2(:,1));
xlabel( ' 时间 ' ); ylabel( ' 摆角 ' );
legend( ' 小摆角 ',' 大摆角 ' );
```

```
    %%程序第三部分计算最大摆角与周期的关系
theta=linspace(pi/360,pi/2,20);
T=[ ];
options=odeset('Events','on');      %%开启事件判断功能
for i=1:20 ;
   [t,u,event]=ode45('djddbfun',[0:0.001:20],[theta(i),0],options);
   T=[T,2*t(end)];
end

figure
plot(theta,T)
title( ' 周期与摆角的关系 ' );
xlabel( ' 摆角 ' );
ylabel( ' 周期 ' );
```

函数文件是一个独立的文件，文件名为 djddbfun.m 。

```
function varargout=djddbfun(t,y,flag)
switch flag
case' '
   varargout{1}=f(t,y);
case 'events'
   [varargout{1:3}]=events(t,y);
otherwise
   error(['unknown flag' '' flag '' '.']);
end

function ydot=f(t,y)
ydot=[y(2);
     -9.8*sin(y(1))];

function [value,isterminal,direction]=events(t,y)
value=y(2);
isterminal=1;
direction=1;
```

2.12 弹簧摆

1. 实验题目

如图 2.20 所示，设质量为 m 的摆锤挂在劲度系数为 k，原长为 l_0 的轻弹簧上，弹簧的另一端悬挂于固定点 O，系统静止自然下垂时弹簧长度为 $l=l_0+\frac{mg}{k}$，系统可在过 O 点的竖直平面内自由摆动，试研究摆锤的运动。

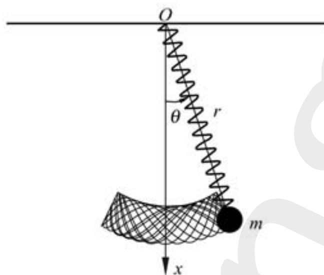


图 2.20 弹簧摆的运动

2. 实验目的和要求

(1) 作出弹簧摆的模拟运动图像并画出摆锤的运动轨迹。

(2) 学习一些作模拟动画的技巧，如在模拟摆锤运动的同时画出摆锤的运动轨迹，用正弦曲线表示弹簧，并且通过极坐标变换来表示运动中的弹簧。

3. 解题分析

系统自由度为 2。以 O 为极点，竖直向下的 Ox 轴为极轴，建立极坐标系，如图 2.20 所示。r 为质点 m 到 O 点距离，θ 为 Ox 轴与弹簧间的夹角。则系统的拉格朗日函数为

$$L=T-V=\frac{1}{2}m\left[\left(\frac{\mathrm{d}r}{\mathrm{d}t}\right)^2+r^2\left(\frac{\mathrm{d}\theta}{\mathrm{d}t}\right)^2\right]-\left[-mgr\cos\theta+\frac{1}{2}k\left(r-l_0\right)^2\right] \tag{2.12.1}$$

由拉格朗日方程可求出系统的运动微分方程为

$$\left.\begin{aligned}\frac{\mathrm{d}^2r}{\mathrm{d}t^2}&=r\left(\frac{\mathrm{d}\theta}{\mathrm{d}t}\right)^2+g\cos\theta-\frac{k}{m}\left(r-l+\frac{mg}{k}\right)\\\frac{\mathrm{d}^2\theta}{\mathrm{d}t^2}&=-\frac{2}{r}\frac{\mathrm{d}r}{\mathrm{d}t}\frac{\mathrm{d}\theta}{\mathrm{d}t}-\frac{g}{r}\sin\theta\end{aligned}\right\} \tag{2.12.2}$$

上式未作小摆角近似，因此可用以研究弹簧摆的大摆角运动，但是难以求出解析解。由数值计算结果所画出弹簧摆的大摆角运动轨迹，可发现它的运动情况很复杂，不作数值计算是无法想象的。

令 $y_1=r, y_2=\dfrac{\mathrm{d}r}{\mathrm{d}t}, y_3=\theta, y_4=\dfrac{\mathrm{d}\theta}{\mathrm{d}t}$ ，则 (2.12.2) 式成为

$$\left.\begin{aligned}
&\frac{\mathrm{d}y_1}{\mathrm{d}t}=y_2\\
&\frac{\mathrm{d}y_2}{\mathrm{d}t}=y_1y_4^2+g\cos y_3-\frac{k}{m}\left(y_1-l+\frac{mg}{k}\right)\\
&\frac{\mathrm{d}y_3}{\mathrm{d}t}=y_4\\
&\frac{\mathrm{d}y_4}{\mathrm{d}t}=-\frac{2}{r}y_2y_4-\frac{g}{y_1}\sin y_3
\end{aligned}\right\}$$

程序中解微分方程的过程比较简单，主要的技巧是在作模拟动画上。有两点值得注意，一是模拟弹簧的运动，另一点是在模拟弹簧运动的同时画出轨迹图。程序运行的结果如图 2.22 所示。

4. 思考题

(1) 能不能找到其它方法来表示运动中的弹簧，试一试。并比较你的方法与本书介绍的方法的优劣。

(2) 在小摆角近似的条件下，方程能否有解析解？试加以分析。

5. 参考程序

主程序的文件名是 thb.m 。

```
theta0=pi/10       %%初始角度，可设不同的值
m = 1; k = 80; g = 9.8;
L0 = 1;       %% L0 为弹簧原来长度
L = L0 + m*g/k;       %% L 为弹簧静止时长度
[t,u1] = ode45('thbfun',[0:0.005:15],[L0 0 theta0 0],[ ],L,k,m,g);
[y1,x1] = pol2cart(u1(:,3),u1(:,1));       %%将极坐标换为直角坐标
y1 = -y1;

figure
ymax = max(abs(y1));
axis([-1.2  1.2  -1.2*ymax  0.2]);      %%设置坐标范围
axis off
title( ' 弹簧摆 ', 'fontsize', 14)
hold on;
R =0.055 ;       %%设置弹簧半径
yy = -L0:0.01:0;
```

```
xx = R*sin(yy./L0*30*pi);     %%用正弦曲线表示垂直位置的弹簧
[a,r] = cart2pol(xx,yy);      %%将弹簧直角坐标换成极坐标
a = a + theta0;     %%通过极角的转动将垂直位置的弹簧转到初始位置
[xx,yy] = pol2cart(a,r);     %%将初始位置弹簧的极坐标换回直角坐标
line([-1 1],[0 0],'color','r','linewidth',2)     %%画横杆
ball = line(xx(1),yy(1),'color','r','marker','.',...
   'markersize',70,'erasemode','xor');     %%画摆球
ball2 = line(xx(1),yy(1),'color',[0.5 0.51 0.6],'linestyle','-',...
   'linewidth',1.3,'erasemode','none');     %%画摆球的运动轨迹
spring = line(xx,yy,'color','g','linewidth',2,...
   'erasemode', 'xor');     %%画弹簧
pause(0.5)

for i = 1 :  length(t)      %%模拟弹簧摆的运动
   yy = -u1(i,1) :  0.01 :  0;
   xx = R*sin(yy./u1(i,1)*30*pi);
   [a,r] = cart2pol(xx,yy);
   a = a + u1(i,3);
   [xx,yy] = pol2cart(a,r);
   set(ball,'XData',x1(i),'YData',y1(i));
   set(ball2,'XData',x1(i),'YData',y1(i));
   set(spring,'XData',xx,'YData',yy);
   drawnow;
end
```

函数文件是一个独立的文件，文件名为 thbfun.m 。

```
function F = thbfun(t,u,flag,l,k,m,g)
F = [u(2);
    u(1)*u(4)^2 + g*cos(u(3)) - k/m*(u(1) - l + m*g/k);
    u(4);
    -2*u(2)*u(4)/u(1) - g*sin(u(3))/u(1)];
```

2.13 滑动摆

1. 实验题目

一个单摆悬挂于可沿水平光滑轨道滑动的滑块上，如图 2.21 所示。滑块质量为 m_1，单摆的杆长为 l，摆锤质量为 m_2，整个系统可在同一竖直平面内运动。试研究整个系统的运动。

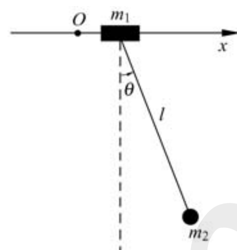


图 2.21 滑动摆的运动

2. 实验目的和要求

(1) 研究滑动摆的运动情况并作出滑动摆的模拟运动图像。

(2) 学习在屏幕上设置图形窗口的不同位置。

3. 解题分析

沿直线水平轨道建立 Ox 轴，以 x 表示滑块在轨道上的位置，以 θ 表示摆杆与竖直线的夹角。则系统的拉格朗日函数为

$$L = T - V = \frac{1}{2}(m_1 + m_2)\left(\frac{\mathrm{d}x}{\mathrm{d}t}\right)^2 + \frac{1}{2}m_2 l^2\left(\frac{\mathrm{d}\theta}{\mathrm{d}t}\right)^2 + m_2 l\frac{\mathrm{d}x}{\mathrm{d}t}\frac{\mathrm{d}\theta}{\mathrm{d}t}\cos\theta + m_2 gl\cos\theta \tag{2.13.1}$$

令 $M = \dfrac{m_2}{m_1 + m_2}$，由拉格朗日方程得系统的运动微分方程

$$\left.\begin{aligned}\frac{\mathrm{d}^2\theta}{\mathrm{d}t^2} &= \frac{-M\cos\theta\sin\theta\left(\dfrac{\mathrm{d}\theta}{\mathrm{d}t}\right)^2 - \dfrac{g}{l}\sin\theta}{1 - M\cos^2\theta}\\ \frac{\mathrm{d}^2x}{\mathrm{d}t^2} &= \frac{Mg\cos\theta\sin\theta + Ml\sin\theta\left(\dfrac{\mathrm{d}\theta}{\mathrm{d}t}\right)^2}{1 - M\cos^2\theta}\end{aligned}\right\} \tag{2.13.2}$$

上式未作小摆角近似，可研究大摆角运动情况。在小摆角情况下，$\sin\theta \approx \theta$，$\cos\theta \approx 1$，$\sin\theta\mathrm{d}\theta/\mathrm{d}t$ 项为高阶项可略去，可得线性化的微振动方程

$$\left.\begin{aligned}\frac{\mathrm{d}^2\theta}{\mathrm{d}t^2}&=-\frac{m_1+m_2}{m_1}\frac{g}{l}\theta\\\frac{\mathrm{d}^2x}{\mathrm{d}t^2}&=\frac{m_2}{m_1}g\theta\end{aligned}\right\}$$

令 $y_1=\theta, y_2=\dfrac{\mathrm{d}\theta}{\mathrm{d}t}, y_3=x, y_4=\dfrac{\mathrm{d}x}{\mathrm{d}t}$ 则 (2.13.2) 式化为四个一阶微分方程

$$\left.\begin{aligned}\frac{\mathrm{d}y_1}{\mathrm{d}t}&=y_2\\\frac{\mathrm{d}y_2}{\mathrm{d}t}&=\frac{-My_2^2\cos y_1\sin y_1-\dfrac{g}{l}\sin y_1}{1-M\cos^2 y_1}\\\frac{\mathrm{d}y_3}{\mathrm{d}t}&=y_4\\\frac{\mathrm{d}y_4}{\mathrm{d}t}&=\frac{Mg\cos y_1\sin y_1+Mly_2^2\sin y_1}{1-M\cos^2 y_1}\end{aligned}\right\}$$

在程序中为了使滑动摆的模拟运动图和滑块与摆锤的位移曲线图能出现在监视器屏幕上的不同位置，用指令 set 对图形窗口的位置进行了设置。在该语句中，指令 gcf 表示获取当前图形窗口的句柄，指令 units 是设置位置数据的单位，指令 normalized 表示使用归一化坐标，即屏幕的左下角为 [0, 0], 屏幕的右上角为 [1, 1] 。指令 position 是用坐标表示图形窗口的位置，其后的 4 个数据分别表示图形窗口左下角的横坐标与纵坐标、图形窗口的宽与高。程序运行结果如图 2.22 所示。

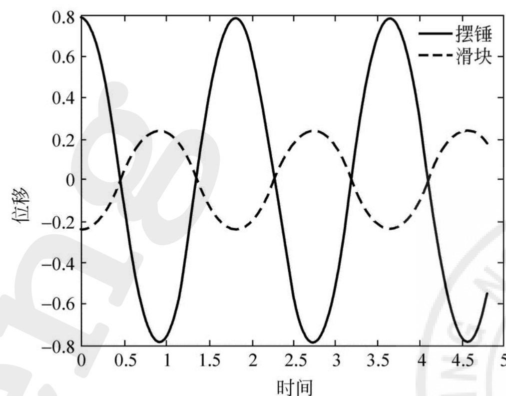


图 2.22 滑块与摆锤的位移曲线

4. 思考题

按照以下情况，对于滑动摆设置不同的初始条件：

(1) 滑块与摆锤的位移不为零，但初速度都为零；

(2) 滑块与摆锤的位移为零，但滑块速度不为零，摆锤的速度为零；

(3) 滑块与摆锤的位移为零，但滑块的速度为零，摆锤的速度为零。

观察滑动摆的运动有何不同，同时注意滑块及摆锤的初始位置对运动有什么影响？注意此时在程序中要重新设置坐标轴的范围及横杆的长度。

5. 参考程序

主程序的文件名是 hdb.m 。

```
g=9.8;m1=4;m2=2;l=1;
[t,y]=ode45('hdbfun',[0:0.001:5],[pi/4,0,...
   -l*cos(pi/4)*2/(4+2),0],[ ], m1, m2, g, l);
figure(1)     %%下面先设定图形窗口的位置
set(gcf,'unit','normalized','position',[0.03 0.1 0.5 0.5]);
cla;
plot(t,y(:,1),t,y(:,3))     %%位移图形
xlabel( ' 时间 ' );
ylabel( ' 位移 ' );
legend( ' 摆锤 ', ' 滑块 ' );

figure(2)
set(gcf,'unit','normalized','position',[0.5 0.4 0.5 0.5]);
cla;
axis([-0.6 0.6 -1 0.2]);
axis off
axis equal
hold on
y1=-l.*cos(y(:,1)); x1=y(:,3)+l.*sin(y(:,1));      %%球的坐标
xian1=line([-0.6,0.6],[0,0],'linewidth',2);     %%画横线
xian2=line([0,0],[0,-1],'linewidth',2,'linestyle',':');      %%画竖线
    %%画杆，滑块与球的初位置
gan=line([y(1,3),x1(1)],[0,y1(1)],'color','y','linestyle','-',...
    'linewidth', 3, 'erasemode','xor');
kuai = line([y(1,3)-0.05,y(1,3)+0.05],[0,0],'color', 'b', ...
    'linestyle', '-', 'linewidth', 15, 'erasemode','xor');
qiu=line(x1(1),y1(1),'color','r','marker','.','markersize',...
    50,'erasemode', 'xor');
```

```
for i=1:5001        %% 作动画
    set(gan,'xdata',[y(i,3),x1(i)],'ydata',[0,y1(i)]);
    set(kuai,'xdata',[y(i,3)-0.05,y(i,3)+0.05],'ydata',[0,0]);
    set(qiu,'xdata',x1(i),'ydata',y1(i));
    drawnow;
end
```

函数文件是一个独立的文件，文件名为 hdbfun.m 。

```
function ydot=hdbfun(t,y,flag,m1,m2,g,l)
M=m2/(m1+m2);
ydot=[y(2);
     (-M*sin(y(1))*cos(y(1))*y(2)^2-...
  g/l*sin(y(1))))/(1-M*cos(y(1))^2);
     y(4);
     (M*g*sin(y(1))*cos(y(1))+M*l*...
  sin(y(1))*y(2)^2)/(1-M*cos(y(1))^2)];
```



2.14 傅科摆

1. 实验题目

法国物理学家傅科于 1851 年在巴黎万圣殿内的拱顶上悬挂了一个摆长 67 m，摆锤质量为 28 kg 的单摆。该单摆摆动周期约为 16 s 。实验发现该摆摆平面绕竖直轴作顺时针转动（由上而下看），转动周期约 32 h，这就是著名的傅科摆实验。这个实验无需依赖地球以外的物体，就能直观地展示地球自转的存在，因此至今仍受到重视。本题目研究傅科摆摆锤在水平面内的轨迹。设摆长为 l，摆锤质量为 m，悬挂于北纬 λ 处。

2. 实验目的和要求

(1) 研究傅科摆在水平面内的运动并画出摆锤在水平面内的运动轨迹。

(2) 研究纬度，初始条件对傅科摆运动轨迹的影响。

(3) 学习指令 input 的用法，学会在程序运行中从键盘向程序输入参数。

3. 解题分析

由于摆长很长，当摆作小角度摆动时，可认为摆锤在水平面内运动。以摆锤平衡位置为原点 O， Ox 指向正南面， Oy 指向正东面，建立直角坐标系。摆锤受重力、科氏力和摆线张力 $\boldsymbol{F}_T$ 作用。与实验 2.8 落体偏东比较，摆锤受力仅多出摆线张力 $\boldsymbol{F}_T$，忽略摆锤沿竖直方向的运动，且知 $F_T \approx mg$，可得出运动微分方程为

$$\left.\begin{aligned}\frac{\mathrm{d}^2x}{\mathrm{d}t^2}-2\omega\frac{\mathrm{d}y}{\mathrm{d}t}\sin\lambda+\frac{g}{l}x=0\\ \frac{\mathrm{d}^2y}{\mathrm{d}t^2}+2\omega\frac{\mathrm{d}x}{\mathrm{d}t}\sin\lambda+\frac{g}{l}y=0\end{aligned}\right\}\tag{2.14.1}$$

这里 g 取 9.8， l 取 67， ω 为地球自转角速度。该方程为线性微分方程组，可求解析解并分析摆锤的运动，但求解过程较为烦琐。

数值求解该方程的程序极简单，充分体现了 MATLAB 的优点，在此不再叙述。注意在程序中为了使摆平面转动效果较为明显，适当放大了地球自转角速度。为了研究纬度及初始条件对摆锤运动轨迹的影响，运用了指令 input 。在程序运行后，根据指令窗口的提示，先输入纬度值，范围可在 $-90\sim90$ 。由于纬度是以角度为单位，故在程序中要将它换算为弧度。然后再依次输入在初始时刻摆锤在 x 方向的位置与速度，在 y 方向的位置与速度，并将这 4 个数用方括号括起来。输入时可选择以下几种情况，如初始位置不为零但初始速度为零，初始位置为零但初始速度不为零，以及初始位置及初始速度都不为零。然后观

察摆锤运动轨迹的差别。图 2.23 是 $\lambda = 30$， $[\,x0, vx0, y0, vy0\,] = [\,4, 0, 0, 0\,]$ 时，程序运行的结果。

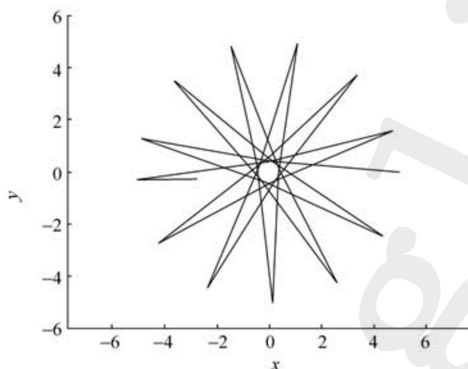


图 2.23 傅科摆的运动轨迹

4. *思考题*

傅科摆实验能否做成三维的模拟动画？试试看。

5. *参考程序*

主程序的文件名是 fkb.m 。

```
a=input(' 请输入纬度 =');
q=input(' 请按此格式依次输入 [x0,vx0,y0,vy0]=');
c=a*pi/180;
[t,x]=ode45('fkbfun',[0:0.02:100],q,[ ],c);
xlabel('x');
ylabel('y');
comet(x(:,1),x(:,3))
```

函数文件是一个独立的文件，文件名为 fkbfun.m 。

```
function tt=fkb(t,x,flag,c)
a=(2*pi*sin(c))/100;
b=9.8/67;
tt=[x(2);
   2*a*x(4)-b*x(1);
   x(4);
   -2*a*x(2)-b*x(3)];
```

2.15　铰链连接的双摆

1. 实验题目

两个摆长为 l 摆锤质量为 m 的单摆，用光滑铰链连接成双摆，如图 2.24 所示。两个摆只能在同一竖直平面内运动。研究它们的运动。

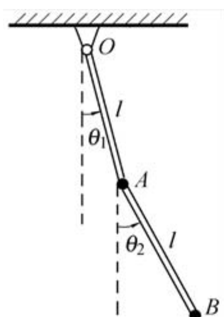


图 2.24　铰链连接的双摆

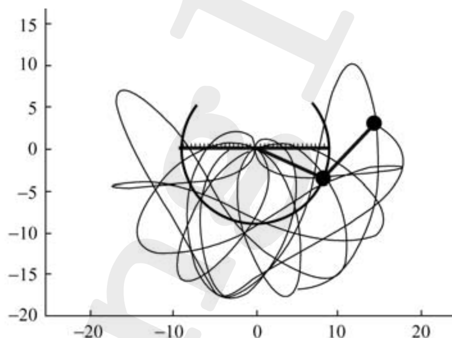


图 2.25　铰链双摆的运动

2. 实验目的和要求

画出双摆的位移曲线并用动画模拟它们的运动。

3. 解题分析

以摆锤偏离平衡位置的摆角 θ_1 和 θ_2 为广义坐标，系统的拉格朗日函数为

$$L = ml^2\left(\frac{\mathrm{d}^2\theta_1}{\mathrm{d}t^2}\right)+\frac{1}{2}ml^2\left(\frac{\mathrm{d}^2\theta_2}{\mathrm{d}t^2}\right)+ml^2\frac{\mathrm{d}\theta_1}{\mathrm{d}t}\frac{\mathrm{d}\theta_2}{\mathrm{d}t}\cos(\theta_2-\theta_1)+ mgl(2\cos\theta_1+\cos\theta_2) \tag{2.15.1}$$

由拉格朗日方程得

$$2l\frac{\mathrm{d}^2\theta_1}{\mathrm{d}t^2}+l\frac{\mathrm{d}^2\theta_2}{\mathrm{d}t^2}\cos(\theta_2-\theta_1)-l\left(\frac{\mathrm{d}\theta_2}{\mathrm{d}t}\right)^2\sin(\theta_2-\theta_1)+2g\sin\theta_1=0 \tag{2.15.2}$$

$$l\frac{\mathrm{d}^2\theta_2}{\mathrm{d}t^2}+l\frac{\mathrm{d}^2\theta_1}{\mathrm{d}t^2}\cos(\theta_2-\theta_1)+l\left(\frac{\mathrm{d}\theta_1}{\mathrm{d}t}\right)^2\sin(\theta_2-\theta_1)+g\sin\theta_2=0 \tag{2.15.3}$$

由方程 (2.15.3) 得

$$l\frac{\mathrm{d}^2\theta_2}{\mathrm{d}t^2}=-l\frac{\mathrm{d}^2\theta_1}{\mathrm{d}t^2}\cos(\theta_2-\theta_1)-l\left(\frac{\mathrm{d}\theta_1}{\mathrm{d}t}\right)^2\sin(\theta_2-\theta_1)-g\sin\theta_2 \tag{2.15.4}$$

将方程 (2.15.4) 代入方程 (2.15.2) 得

$$\begin{aligned}\frac{\mathrm{d}^2\theta_1}{\mathrm{d}t^2} &= \frac{1}{2l-l\cos^2(\theta_2-\theta_1)}\left[l\left(\frac{\mathrm{d}\theta_1}{\mathrm{d}t}\right)^2\sin(\theta_2-\theta_1)\cos(\theta_2-\theta_1)+\right.\\ &\left. g\sin\theta_2\cos(\theta_2-\theta_1)+l\left(\frac{\mathrm{d}\theta_2}{\mathrm{d}t}\right)^2\sin(\theta_2-\theta_1)-2g\sin\theta_1\right] \end{aligned} \quad (2.15.5)$$

由方程 (2.15.2) 得

$$l\frac{\mathrm{d}^2\theta_1}{\mathrm{d}t^2} = -\frac{1}{2}l\frac{\mathrm{d}^2\theta_2}{\mathrm{d}t^2}\cos(\theta_2-\theta_1)+\frac{1}{2}l\left(\frac{\mathrm{d}\theta_2}{\mathrm{d}t}\right)^2\sin(\theta_2-\theta_1)-g\sin\theta_1 \quad (2.15.6)$$

将方程 (2.15.6) 代入方程 (2.15.3) 得

$$\begin{aligned}\frac{\mathrm{d}^2\theta_2}{\mathrm{d}t^2} &= \frac{1}{2l-l\cos^2(\theta_2-\theta_1)}\left[-l\left(\frac{\mathrm{d}\theta_2}{\mathrm{d}t}\right)^2\sin(\theta_2-\theta_1)\cos(\theta_2-\theta_1)+\right.\\ &\left. 2g\sin\theta_1\cos(\theta_2-\theta_1)-2l\left(\frac{\mathrm{d}\theta_1}{\mathrm{d}t}\right)^2\sin(\theta_2-\theta_1)-2g\sin\theta_2\right] \end{aligned} \quad (2.15.7)$$

令 $y_1=\theta_1, y_2=\frac{\mathrm{d}\theta_1}{\mathrm{d}t}, y_3=\theta_2, y_4=\frac{\mathrm{d}\theta_2}{\mathrm{d}t}$，由方程 (2.15.5) 和方程 (2.15.7) 得

$$\left.\begin{aligned}\frac{\mathrm{d}y_1}{\mathrm{d}t} &= y_2\\ \frac{\mathrm{d}y_2}{\mathrm{d}t} &= \frac{1}{2l-l\cos^2(y_3-y_1)}\left[ly_2^2\sin(y_3-y_1)\cos(y_3-y_1)+\right.\\ &\left. g\sin y_3\cos(y_3-y_1)+ly_4^2\sin(y_3-y_1)-2g\sin y_1\right]\\ \frac{\mathrm{d}y_3}{\mathrm{d}t} &= y_4\\ \frac{\mathrm{d}y_4}{\mathrm{d}t} &= \frac{1}{2l-l\cos^2(y_3-y_1)}\left[-ly_4^2\sin(y_3-y_1)\cos(y_3-y_1)+\right.\\ &\left. 2g\sin y_1\cos(y_3-y_1)-2ly_2^2\sin(y_3-y_1)-2g\sin y_3\right]\end{aligned}\right\}$$

在程序中仿照实验 2.13 滑动摆的做法，在用动画表示双摆的运动的同时，也在监视器屏幕上显示双摆的位移曲线。至于双摆运动时的轨迹的画法与实验 2.12 的弹簧摆的画法相同。图 2.25 和图 2.26 是程序运行的结果，改变初始条件会得到不同的曲线。

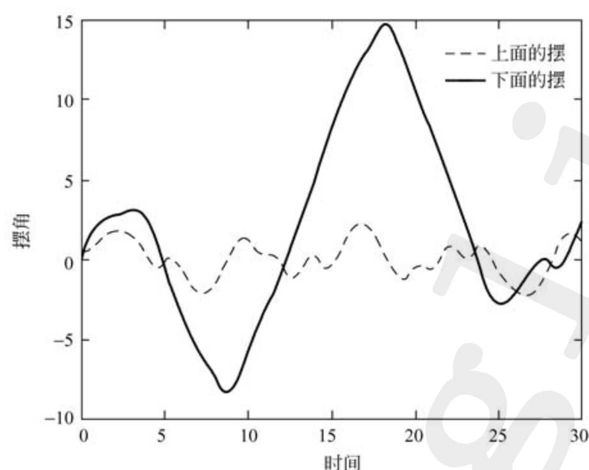


图 2.26 两个摆的位移曲线

4. 思考题

如果希望模拟绞链连接的双摆的微振动下的两种简正模的运动状态，应该如何设置初始条件？通过改变程序中的初始条件后运行程序来验证你的想法。

5. 参考程序

主程序的文件名是 jjsb.m 。

```
l=9;
[t,u]=ode45('jjsbfun',[0:0.01:30],[0.5,0.2,0.1,2.8],[],l);
y1=-l*cos(u(:,1));      %%计算球 1 的坐标
x1=l*sin(u(:,1));
y2=y1-l*cos(u(:,3));      %%计算球 2 的坐标
x2=x1+l*sin(u(:,3));

figure(1)
set(gcf,'unit','normalized','position',...
   [0.03 0.1 0.5 0.5]);      %%设置窗口坐标
cla;
plot(t,u(:,1),'g',t,u(:,3),'r')      %%画位移曲线
xlabel(' 时间 ');ylabel(' 摆角 ');
legend(' 上面的摆 ',' 下面的摆 ');
pause(0.5)
```

```
figure(2)
set(gcf,'unit','normalized','position',...
   [0.5 0.4 0.5 0.5]);     %%设置窗口坐标
cla;
axis([-10 10 -20 20])
axis equal
hold on
a10=line([-9,9],[0,0],'color','k','linewidth',3.5);     %%画横梁
    %%下面的循环语句是画横梁顶部的虚线
a20=linspace(-9,9,36);
for i=1:35
   a30=(a20(i)+a20(i+1))/2;
   plot([a20(i),a30],[0,0+0.5],'color','b',...
      'linestyle','-','linewidth',1);
end
    %%以下为模拟动画
ball1a=line(x1(1),y1(1),'color',[0.5 0.6 0.4],'linestyle',...
   '-','linewidth',1,'erasemode','none');
ball1=line(x1(1),y1(1),'color','r','marker','.',...
   'markersize',40,'erasemode','xor');
ball2a=line(x2(1),y2(1),'color',[0.5 0.6 0.4],'linestyle',...
   '-','linewidth',1,'erasemode','none');
ball2=line(x2(1),y2(1),'color','r','marker','.',...
   'markersize',40,'erasemode','xor');
gan1=line([0,x1(1)],[0,y1(1)],'color','g',...
      'linewidth',2,'erasemode','xor');
gan2=line([x1(1),x2(1)],[y1(1),y2(1)],'color','g',...
   'linewidth',2,'erasemode','xor');

for i=1:length(u)
   set(ball1,'xdata',x1(i),'ydata',y1(i));
   set(ball2,'xdata',x2(i),'ydata',y2(i));
   set(ball1a,'xdata',x1(i),'ydata',y1(i));
   set(ball2a,'xdata',x2(i),'ydata',y2(i));
   set(gan1,'xdata',[0,x1(i)],'ydata',[0,y1(i)]);
   set(gan2,'xdata',[x1(i),x2(i)],'ydata',[y1(i),y2(i)]);
```

```
   drawnow
end
```

函数文件是一个独立的文件，文件名为 jjsbfun.m 。

```
function ydot=jjsbfun(t,y,flag,l)
g=9.8;
ydot=[y(2);
     (l*y(2)^2*sin(y(3)-y(1))*cos(y(3)-y(1))...
 +g*sin(y(3))*cos(y(3)-y(1))+ l*y(4)^2*sin(y(3)-y(1))...
        -2*g*sin(y(1)))/ (2*l-l*(cos(y(3)-y(1)))^2);
      y(4);
     (-l*y(4)^2*sin(y(3)-y(1))*cos(y(3)-y(1))+...
 2*g*sin(y(1))*cos(y(3)-y(1))-2*l*y(2)^2*sin(y(3)-y(1))...
        -2*g*sin(y(3)))/(2*l-l*(cos(y(3)-y(1)))^2)];
```




2.16 弹簧连接的耦合摆

1. 实验题目

研究由弹簧耦合的两个相同单摆的运动。耦合摆由两个相同的单摆和一个连接两摆锤的弹簧组成，如图 2.27 所示。弹簧原长 a，等于摆的两个悬挂点之间的距离，倔强系数为 k。摆锤质量为 m，摆杆长为 l，杆的质量忽略不计。设两摆均在同一竖直平面内摆动。

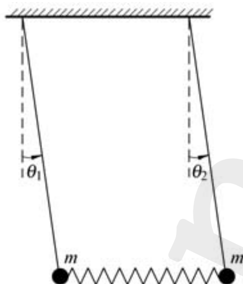


图 2.27 弹簧连接的耦合摆

2. 实验目的和要求

(1) 在系统作小摆角微振动情况下，求系统的两个简正模和一般振动，对结果做动画模拟。

(2) 作出摆的位移曲线，观察耦合摆在一般振动下出现的振幅调制现象。

(3) 学习用矩阵解本征值问题。

3. 解题分析

以摆杆和竖直方向的夹角 θ_1， θ_2 为广义坐标，系统的拉格朗日函数为

$$\begin{aligned}L &= T - V \\ &= \frac{1}{2}ml^2\left[\left(\frac{\mathrm{d}\theta_1}{\mathrm{d}t}\right)^2 + \left(\frac{\mathrm{d}\theta_2}{\mathrm{d}t}\right)^2\right] - \\ &\quad \frac{1}{2}k\left[\sqrt{(a + l\sin\theta_2 - l\sin\theta_1)^2 + (l\cos\theta_2 - l\cos\theta_1)^2} - a\right]^2 + \\ &\quad mgl(\cos\theta_1 + \cos\theta_2)\end{aligned} \tag{2.16.1}$$

当 $\theta_1 \ll 1, \theta_2 \ll 1$ 时，作级数展开

$$L = \frac{1}{2}ml^2\left[\left(\frac{\mathrm{d}\theta_1}{\mathrm{d}t}\right)^2 + \left(\frac{\mathrm{d}\theta_2}{\mathrm{d}t}\right)^2\right]$$

$$-\frac{1}{2}kl^2(\theta_2-\theta_1)^2+mgl\left(2-\frac{\theta_1^2}{2}-\frac{\theta_2^2}{2}\right) \tag{2.16.2}$$

代入拉格朗日方程得到系统微振动的运动微分方程为

$$\left.\begin{aligned} ml\frac{\mathrm{d}^2\theta_1}{\mathrm{d}t^2}+kl\theta_1+mg\theta_1-kl\theta_2=0 \\ ml\frac{\mathrm{d}^2\theta_2}{\mathrm{d}t^2}+kl\theta_2+mg\theta_2-kl\theta_1=0 \end{aligned}\right\} \tag{2.16.3}$$

令

$$\theta_1=A_1\cos(\omega t+\varphi),\quad \theta_2=A_2\cos(\omega t+\varphi)$$

代入上式可求出系统的简正频率为

$$\left.\begin{aligned} \omega_1&=\sqrt{\frac{g}{l}} \\ \omega_2&=\sqrt{\frac{g}{l}+\frac{2k}{m}} \end{aligned}\right\} \tag{2.16.4}$$

与 ω_1 相对应的简正振动方程为

$$\left.\begin{aligned} \theta_1&=A_{11}\cos(\omega_1 t+\varphi_1) \\ \theta_2&=A_{11}\cos(\omega_1 t+\varphi_1) \end{aligned}\right\} \tag{2.16.5}$$

与 ω_2 相对应的简正振动方程为

$$\left.\begin{aligned} \theta_1&=A_{12}\cos(\omega_2 t+\varphi_2) \\ \theta_2&=-A_{12}\cos(\omega_2 t+\varphi_2) \end{aligned}\right\} \tag{2.16.6}$$

(2.16.3) 式的通解（对应耦合摆的一般振动）为

$$\left.\begin{aligned} \theta_1&=A_{11}\cos(\omega_1 t+\varphi_1)+A_{12}\cos(\omega_2 t+\varphi_2) \\ \theta_2&=A_{11}\cos(\omega_1 t+\varphi_1)-A_{12}\cos(\omega_2 t+\varphi_2) \end{aligned}\right\} \tag{2.16.7}$$

由 (2.16.7) 式可见，每个单摆都同时参与两个简谐振动，合振动情况较为复杂，下面我们在一个特定的初始条件下作一些解析的讨论。

如果初始时，两个摆的初速均为零，一个摆处于平衡位置，另一个摆偏离平衡位置，即 $t=0$ 时，$\theta_1=\theta_{10}$，$\theta_2=0$，$\dfrac{\mathrm{d}\theta_1}{\mathrm{d}t}=\dfrac{\mathrm{d}\theta_2}{\mathrm{d}t}=0$，则

$$\varphi_1=\varphi_2=0$$

$$A_{11}=A_{12}=\frac{1}{2}\theta_0$$

(2.16.7) 式变成

$$\left.\begin{aligned}\theta_1 &= \frac{1}{2}\theta_0\left(\cos\omega_1 t + \cos\omega_2 t\right)\\ \theta_2 &= \frac{1}{2}\theta_0(\cos\omega_1 t - \cos\omega_2 t)\end{aligned}\right\} \tag{2.16.8}$$

若令

$$\omega_a = \frac{\omega_1 + \omega_2}{2}, \quad \omega_b = \frac{\omega_2 - \omega_1}{2}$$

则 (2.16.8) 式可写成

$$\left.\begin{aligned}\theta_1 &= (\theta_0 \cos\omega_b t)\cos\omega_a t\\ \theta_2 &= (\theta_0 \sin\omega_b t)\sin\omega_a t\end{aligned}\right\} \tag{2.16.9}$$

这种情况下每个摆均发生低频 (ω_b) 振动对高频 (ω_a) 振动的调制现象；当 $k \ll m$ 时，ω_1 与 ω_2 相差微小，则 $\omega_a \gg \omega_b$，会产生“拍”的现象。“拍”的现象在普通物理力学中均有讨论，现在我们可以借助位移曲线和运动模拟图像，对“拍”的现象有一个“直观”的了解。图 2.28 就显示了这种“拍”的现象。

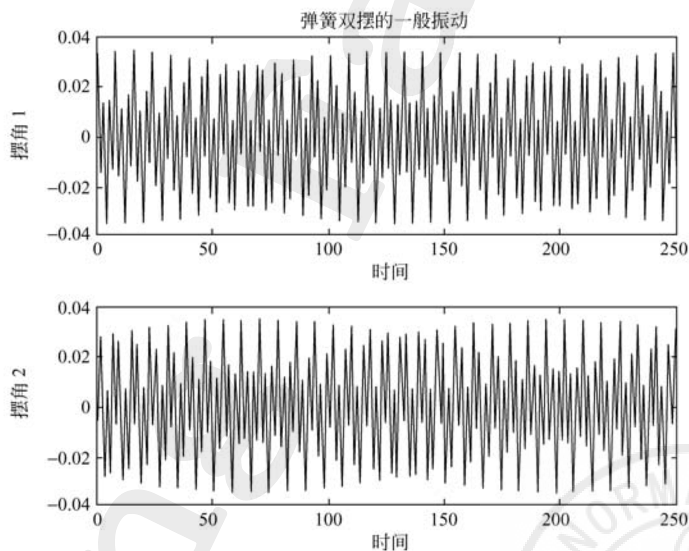


图 2.28 弹簧连接的耦合摆的位移曲线

现在用 MATALB 的矩阵运算功能来解方程 (2.16.3), 令

$$\boldsymbol{P} = \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix}; \quad \boldsymbol{S} = \begin{bmatrix} \frac{k}{m} + \frac{g}{l} & -\frac{k}{m} \\ -\frac{k}{m} & \frac{k}{m} + \frac{g}{l} \end{bmatrix}; \quad \boldsymbol{X} = \begin{bmatrix} \theta_1 \\ \theta_2 \end{bmatrix}$$

则方程 (2.16.3) 可写为

$$\boldsymbol{P}\frac{\mathrm{d}^2}{\mathrm{d}t^2}\boldsymbol{X}+\boldsymbol{S}\boldsymbol{X}=0 \tag{2.16.10}$$

求出矩阵 $\boldsymbol{S}-\lambda\boldsymbol{P}$ 的本征值 L 和本征矢量 $\boldsymbol{M}$，则 L 和 $\boldsymbol{M}$ 满足

$$\boldsymbol{M}'*\boldsymbol{P}*\boldsymbol{M}=\begin{bmatrix}1 & 0\\ 0 & 1\end{bmatrix};\quad L=\boldsymbol{M}'*\boldsymbol{S}*\boldsymbol{M}=\begin{bmatrix}\lambda_1 & 0\\ 0 & \lambda_2\end{bmatrix}$$

考虑到 $\boldsymbol{M}*\boldsymbol{M}'$ 是对角矩阵，得到

$$\boldsymbol{M}'*\boldsymbol{P}*\boldsymbol{M}*\boldsymbol{M}'*\frac{\mathrm{d}^2}{\mathrm{d}t^2}\boldsymbol{X}+\boldsymbol{M}'*\boldsymbol{S}*\boldsymbol{M}*\boldsymbol{M}'*\boldsymbol{X}=0$$

令

$$\boldsymbol{Z}=\begin{bmatrix}z_1\\ z_2\end{bmatrix}=\boldsymbol{M}'*\boldsymbol{X} \tag{2.16.11}$$

得到

$$\frac{\mathrm{d}^2}{\mathrm{d}t^2}\boldsymbol{Z}+L*\boldsymbol{Z}=0$$

这是两个独立的方程

$$\frac{\mathrm{d}^2z_1}{\mathrm{d}t^2}+\lambda_1z_1=0 \tag{2.16.12a}$$

$$\frac{\mathrm{d}^2z_2}{\mathrm{d}t^2}+\lambda_2z_2=0 \tag{2.16.12b}$$

令 $\omega_1^2=\lambda_1$; $\omega_2^2=\lambda_2$ 则方程 (2.16.12a) 和方程 (2.16.12b) 的解可以写成

$$z_1=A_1\cos(\omega_1t+\varphi_1) \tag{2.16.13a}$$

$$z_2=A_2\cos(\omega_2t+\varphi_2) \tag{2.16.13b}$$

其中的常数 A_1，φ_1，A_2，φ_2 由初始条件决定，而方程 (2.16.10) 的解为式 (2.16.11) 的逆变换

$$\boldsymbol{X}=\mathrm{inv}(\boldsymbol{M}')*\boldsymbol{Z}=\boldsymbol{M}*\boldsymbol{Z} \tag{2.16.14}$$

即是

$$\begin{aligned}\theta_1&=M(1,1)z_1+M(1,2)z_2\\&=M(1,1)A_1\cos(\omega_1t+\varphi_1)+M(1,2)A_2\cos(\omega_2t+\varphi_2)\end{aligned} \tag{2.16.15}$$

$$\begin{aligned}\theta_2&=M(2,1)z_1+M(2,2)z_2\\&=M(2,1)A_1\cos(\omega_1t+\varphi_1)+M(2,2)A_2\cos(\omega_2t+\varphi_2)\end{aligned} \tag{2.16.16}$$

以上就是程序中求解弹簧连接的耦合摆微振动的思路,所用的符号也基本相同。考虑到有两种简正模式的振动和一种一般振动,在式(2.16.15)和式(2.16.16)中可以令 $A_1 =[\,0, 0.02, 0.02\,]$, $A_2 = [0.03, 0, -0.03\,]$, 每一组 $A_1(j)$, $A_2(j)$, $j = 1, 2, 3$ 代表一种振动模式，然后将这些结果做成动画。为了显示图 2.28 中的“拍”，需要适当加长解方程的作动画的时间，读者可以自行改变程序中有关数值。

4. 思考题

改变 A_1, A_2 的值，会对“拍”产生什么影响？

5. 参考程序

主程序的文件名是 thsb.m 。

```
l=15; g=9.8; m=80; k=200; h=0.6;
S=[g/l+k/m, -k/m; -k/m, g/l+k/m];
P=[1, 0; 0, 1];
[M,L]=eig(S,P)       %%求本征矢量和本征值
OL=sqrt(L)       %%求本征频率
A1=[0;0.02;0.02]; A2=[0.03;0;-0.03]; phi1=0; phi2=0;
t=0:0.04:150;
str{1}=' 弹簧双摆的反相振动 -- 简正模 1';
str{2}=' 弹簧双摆的同相振动 -- 简正模 2';
str{3}=' 弹簧双摆的一般振动 ';

for j=1:3       %%三种不同的振动情况
   theta1=M(1,1)*A1(j)*cos(OL(1,1)*t+phi1)...
     +M(1,2)*A2(j)*cos(OL(2,2)*t+phi2);
   theta2=M(2,1)*A1(j)*cos(OL(1,1)*t+phi1)...
     +M(2,2)*A2(j)*cos(OL(2,2)*t+phi2);

   figure(1)
   set(gcf,'unit','normalized','position',[0.03 0.1 0.5 0.5]);
   cla;
   subplot(2,1,1)
   axis([0 20 -0.3 0.3])
   plot(t,theta1)
   xlabel(' 时间 ');
   ylabel(' 摆角 1');
```

```
    title(str {j });
    subplot(2,1,2)
    axis([0 20 -0.3 0.3])
    plot(t,theta2)
    xlabel(' 时间 ');
    ylabel(' 摆角 2');
    pause(0.5)

    figure(2)
    set(gcf,'unit','normalized','position',[0.5 0.45 0.5 0.5]);
    cla;
    axis([-15 15 -8 10])
    title(str{j});
    hold on

     %%画横梁和小斜线
    a10=line([-9,9],[9,9],'color','k','linestyle','-','linewidth',3.5);
    a20=linspace(-9,9,36);
    for i=1:35
       a30=(a20(i)+a20(i+1))/2;
       plot([a20(i),a30],[9,9.5],'color','b',...
          'linestyle','-','linewidth',1);
    end
     %%计算两小球的直角坐标
x1=-5+l*sin(theta1);
y1=9-l*cos(theta1);
x2=5+l*sin(theta2);
y2=9-l*cos(theta2);

     %%用正弦函数画弹簧
a1=linspace(x1(1),x2(1),220);
b1=0.7*sin((a1-x1(1))*40/(x2(1)-x1(1)))+y2(1);
tan1=line(a1,b1,'color','m','linestyle','-',...
   'erasemode','xor','linewidth',1.5);
yuan1=line([-5,x1(1)],[9,y1(1)],'color','b','erasemode','xor',...
   'linestyle','-','linewidth',2.5);
```

```
yuan2=line([5,x2(1)],[9,y2(1)],'color','b','erasemode','xor',...
   'linestyle','-','linewidth',2.5);
qiu1=line(x1(1),y1(1),'color','g','erasemode','xor',...
   'marker','.','markersize', 60);
qiu2=line(x2(1),y2(1),'color','g','erasemode','xor',...
   'marker','.','markersize', 60);

n=length(t);
   for i=1:n
      set(yuan1,'xdata',[-5,x1(i)],'ydata',[9,y1(i)]);
      set(yuan2,'xdata',[5,x2(i)],'ydata',[9,y2(i)]);
      set(qiu1,'xdata',x1(i),'ydata',y1(i));
      set(qiu2,'xdata',x2(i),'ydata',y2(i));
           %%用正弦函数画弹簧
      a1=linspace(x1(i),x2(i),220);
      b1=0.7*sin((a1-x1(i))*40/(x2(i)-x1(i)))+y1(i);
      set(tan1,'xdata',a1,'ydata',b1);
      drawnow;
   end
end
```



2.17 三自由度系统的微振动

1. 实验题目

研究两个弹簧连接三个质点组成的一维振动系统的运动。如图 2.29 所示，其中弹簧的倔强系数均为 k, 中间的质点的质量为 m_0, 两端质点的质量都为 m 。

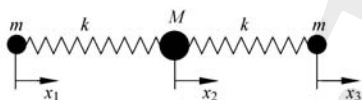


图 2.29 三自由度系统的微振动

2. 实验目的和要求

(1) 用三种不同的方法即矩阵方法、快速傅里叶变换法和拉普拉斯变换法分别求出振动系统的简正频率。从而证明这三种方法所得的结果是相同的。本题将这三种方法放在一起，通过对比，读者可以了解它们之间的内在联系。

(2) 将拉普拉斯变换法得出的系统的运动微分方程的解析解做成动画模拟。

(3) 学习快速傅里叶变换法和拉普拉斯变换法的用法。

3. 解题分析

以图 2.29 中所示的三个质点相对自身平衡位置的位移 x_1, x_2, x_3 作为三自由度振动系统的广义坐标。用拉格朗日方法，可得出系统的运动微分方程：

$$\left.\begin{aligned} &m\frac{\mathrm{d}^2x_1}{\mathrm{d}t^2}+kx_1-kx_2=0\\ &m_0\frac{\mathrm{d}^2x_2}{\mathrm{d}t^2}-kx_1+2kx_2-kx_3=0\\ &m\frac{\mathrm{d}^2x_3}{\mathrm{d}t^2}-kx_2+kx_3=0 \end{aligned}\right\} \tag{2.17.1}$$

方程组的矩阵形式为

$$\boldsymbol{S}\frac{\mathrm{d}^2}{\mathrm{d}t^2}\boldsymbol{X}+\boldsymbol{K}\boldsymbol{X}=0 \tag{2.17.2}$$

其中

$$\boldsymbol{S}=\begin{bmatrix} m & 0 & 0\\ 0 & m_0 & 0\\ 0 & 0 & m \end{bmatrix}\quad \boldsymbol{X}=\begin{bmatrix} x_1\\ x_2\\ x_3 \end{bmatrix}\quad \boldsymbol{K}=\begin{bmatrix} k & -k & 0\\ -k & 2k & k\\ 0 & -k & k \end{bmatrix}$$

设解的形式为

$$\left.\begin{aligned} x_1 &= A_1\cos(\omega t+\varphi) \\ x_2 &= A_2\cos(\omega t+\varphi) \\ x_3 &= A_3\cos(\omega t+\varphi) \end{aligned}\right\} \tag{2.17.3}$$

代入 (2.17.1) 式后，得矩阵形式的方程

$$(\boldsymbol{K}-\boldsymbol{S}\,\omega^2)\boldsymbol{A}=0 \tag{2.17.4}$$

其中

$$\boldsymbol{A}=\begin{bmatrix} A_1 \\ A_2 \\ A_3 \end{bmatrix}$$

(2.17.4) 式即

$$\left.\begin{aligned} &(k-m\omega^2)A_1-kA_2=0 \\ &-kA_1+(2k-M\omega^2)A_2-kA_3=0 \\ &-kA_2+(k-m\omega^2)A_3=0 \end{aligned}\right\} \tag{2.17.5}$$

上面的方程组要求

$$\boldsymbol{K}-\boldsymbol{S}\,\omega^2=0 \tag{2.17.6}$$

即

$$\begin{vmatrix} k-m\omega^2 & -k & 0 \\ -k & 2k-M\omega^2 & -k \\ -k & 0 & k-m\omega^2 \end{vmatrix}=0 \tag{2.17.7}$$

由此得到三个简正频率

$$\left.\begin{aligned} \omega_1 &= \sqrt{\frac{k}{m}} \\ \omega_2 &= 0 \\ \omega_3 &= \sqrt{\frac{k}{m}\left(1+\frac{2m}{M}\right)} \end{aligned}\right\} \tag{2.17.8}$$

将简正频率分别代回 (2.17.5) 式，可得到三个与之对应的本征矢量。

对 $\omega_1=\sqrt{k/m}$，本征矢量为

$$A_1=\begin{bmatrix}A_{11}\\A_{21}\\A_{31}\end{bmatrix}=\begin{bmatrix}A_{11}\\0\\-A_{11}\end{bmatrix} \tag{2.17.9}$$

A_{11}，A_{21}，A_{31} 的右脚码的左边数字为质点的编号，右边数字为简正频率的编号。这三个量分别是与 ω_1 对应的简正模式的三个质点的振幅。简正模式的振动方程为

$$\left.\begin{aligned}x_1&=A_{11}\cos(\omega_1 t+\varphi_1)\\x_2&=0\\x_3&=-A_{11}\cos(\omega_1 t+\varphi_1)\end{aligned}\right\} \tag{2.17.10}$$

对 $\omega_2=0$，本征矢量为

$$A_2=\begin{bmatrix}A_{12}\\A_{22}\\A_{32}\end{bmatrix}=\begin{bmatrix}A_{12}\\A_{12}\\A_{12}\end{bmatrix} \tag{2.17.11}$$

简正振动的方程为

$$\left.\begin{aligned}x_1&=A_{12}\cos\varphi_2\\x_2&=A_{12}\cos\varphi_2\\x_3&=A_{12}\cos\varphi_2\end{aligned}\right\} \tag{2.17.12}$$

各质点位移相同，系统作纯平动。

对 $\omega_3=\sqrt{\dfrac{k}{m}\left(1+\dfrac{2m}{M}\right)}$，本征矢量为

$$A_3=\begin{bmatrix}A_{13}\\A_{23}\\A_{33}\end{bmatrix}=\begin{bmatrix}A_{13}\\-\dfrac{2m}{M}A_{13}\\A_{13}\end{bmatrix} \tag{2.17.13}$$

进而得到第三个简正模式的运动学方程

$$\left.\begin{aligned} x_1 &= A_{13}\cos(\omega_3 t+\varphi_3) \\ x_2 &= -\frac{2m}{M}A_{13}\cos(\omega_3 t+\varphi_3) \\ x_3 &= A_{13}\cos(\omega_3 t+\varphi_3) \end{aligned}\right\} \tag{2.17.14}$$

式 (2.17.10) 、式 (2.17.12) 和式 (2.17.14) 描绘出三个简正模式的运动情况。

系统的运动是三个简正模式运动的线性叠加，方程组 (2.17.1) 的通解为

$$\left.\begin{aligned} x_1 &= A_{11}\cos(\omega_1 t+\varphi_1)+A_{12}\cos\varphi_2+A_{13}(\omega_3 t+\varphi_3) \\ x_2 &= A_{12}\cos\varphi_2-\frac{2m}{M}A_{13}\cos(\omega_3 t+\varphi_3) \\ x_3 &= -A_{11}\cos(\omega_1 t+\varphi_1)+A_{12}\cos\varphi_2+A_{13}(\omega_3 t+\varphi_3) \end{aligned}\right\} \tag{2.17.15}$$

积分常数 A_{11}，A_{12}，A_{13} 和 φ_1，φ_2，φ_3 由初始条件确定。

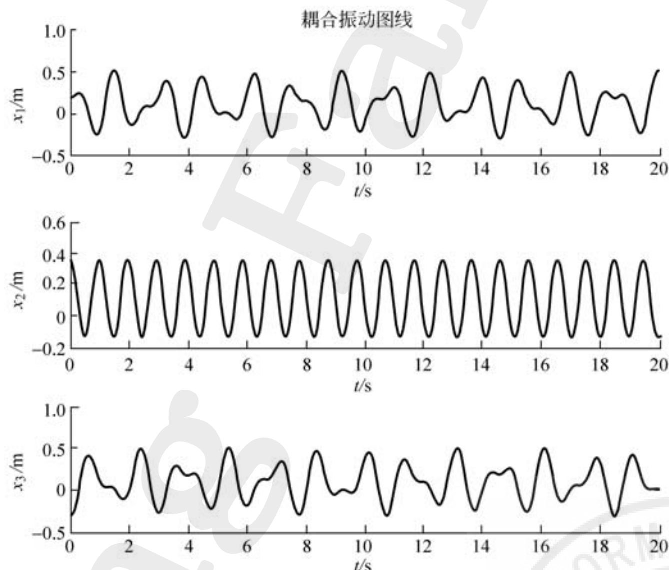


图 2.30 三个质点的位移曲线

在程序中，用矩阵方法求本征频率的方法与 2.16 节弹簧连接的耦合摆中使用的方法相同，这里不再重复。得到的三个本征频率是 6.4550， 4.0825 和 0 。

用傅里叶变换求本征频率的办法是对数值求解微分方程所得到的质点 1 的位移曲线（见图 2.30 ）作傅里叶变换，将得到的频率去掉零频分量和共轭的分

量，然后平方得到功率谱(见图 2.31)。功率谱中两个极大值即是不为零的两个本征频率。得到的结果是 6.4736 和 3.9984。

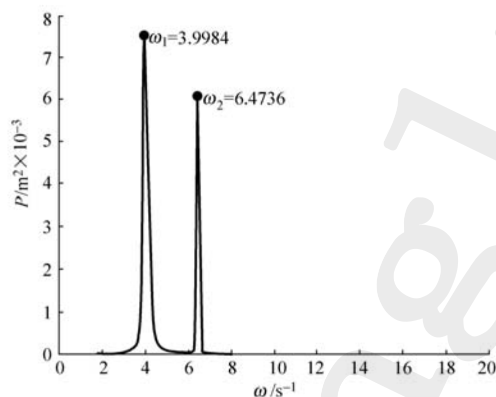


图 2.31 质点 1 的功率谱

用拉普拉斯变换法可求常微分方程的解析解。程序中用 MATLAB 的符号计算功能，对微分方程组进行拉普拉斯变换(为了简单，求解中取质点 1 的初位移为 XL10，其余的初位移和初速度取为零)，然后求解变换所得的方程组，最后将解作逆变换，得出原微分方程组的解。结果与解析方法的解相同。将有关数值代入后结果也相同。在屏幕上显示的分别是

```
XL1 = XL10*m*(1/(M+2*m)+1/2/m*cos((k/m)^(1/2)*t)+
      1/(2*M+4*m)*M/m*cos(((M+2*m)*k/M/m)^(1/2)*t))
XL2 = k*XL10*m*(-1/k/(M+2*m)*cos(((M+2*m)*k/M/m)^(1/2)*t)+
      1/k/(M+2*m))
XL3 = m*XL10*k^2*(1/k^2/(M+2*m)-1/2/k^2/m*cos((k/m)^(1/2)*t)+
      1/2*M/k^2/(M+2*m)/m*cos(((M+2*m)*k/M/m)^(1/2)*t))
```

为了作出动画模拟，必须给参数的具体值。仍取 $m=3, M=4, k=50$, 代入拉普拉斯变换法求得的方程的解析解中，得

```
XL1 = .6000e-1+.1000*cos(4.082*t)+.4000e-1*cos(6.456*t)
XL2 = -.6000e-1*cos(6.456*t)+.6000e-1
XL3 = .6000e-1-.1000*cos(4.082*t)+.4000e-1*cos(6.456*t)
```

这个表达式就是式 (2.17.15)。在每个余弦函数中 ω 前面的数值就是简正频率。仿照 2.16 节的做法，在这个表达式中取不同的项加以组合，就可以分别表示不同的振动模式如一般振动、按简正模式 1 运动、按简正模式 2 运动等。如果没有动画，一般振动模式的运动是很难想象的。

4. 思考题

(1) 傅里叶频率与简正频率是什么关系？

(2) 在利用傅里叶变换求简正频率时，为什么要先求出功率谱？在功率谱中如何寻找它最大值？

(3) 在功率谱图形中，横坐标与纵坐标的单位是什么？

5. 参考程序

主程序的文件名是 szydxt.m 。

```
disp( ' 方法一：用矩阵法求本征值 ' )
m=3, M=4, k=50
K=[k, -k, 0; -k, 2*k, -k; 0, -k, k];
S=[m, 0, 0; 0, M, 0; 0, 0, m];
[Q,L]=eig(K,S)      %%求本征矢量与本征值
OL=sqrt(L)       %%求本征频率
pause(0.2)

disp( ' 方法二：用傅里叶变换求数值解的本征频率 ' )
XS10=0.2;XS20=0.35;XS30=-0.3;       %%初始位移
    %%求数值解
[t,u]=ode45('szydxtfun',[0:0.01:33],[XS10,XS20,XS30,0,0,0],[ ]);
figure       %%画三个质点的位移曲线
subplot(3,1,1);
plot(t(1:2000),u(1:2000,1))
title(' 耦合振动图线 ');
xlabel('Time(s)');    ylabel('Distance(m)');
subplot(3,1,2);
plot(t(1:2000),u(1:2000,2))
xlabel('Time(s)');    ylabel('Distance(m)');
subplot(3,1,3);
plot(t(1:2000),u(1:2000,3))
xlabel('Time(s)');    ylabel('Distance(m)');
    %%用傅里叶变换求本征频率
Y=fft(u(:,1));       %%对数值解作傅里叶变换
Y(1)=[];      %%去掉零频分量
n=length(Y)/2;      %%计算频率个数
power=abs(Y(1:n)).^2/(length(Y).^2);      %%计算功率谱
```

```
freq=100*(1:n)/length(Y);
    %%计算频率，因为步长为 0.01 ，而不是 1 ，故乘以 100
power1=power;     %%寻找两个最大的频率
[id1,daa1]=max(power1);
power1(daa1)=0;
[id2,daa2]=max(power1);
WF=2*pi*[freq(daa1),freq(daa2)];     %%求圆频率
figure
plot(2*pi*freq(1:100),power(1:100))     %%画功率谱图
hold on
title('功率谱')
xlabel('\omega /s^{-1}')    ylabel('P / m^2')
    %%画标志点
plot(WF,[power(daa1),power(daa2)],'r.','markersize',40)
text(5,7.5e-3,'Frequency1=3.9984')     %%加文字注释
text(7,6e-3,'Frequency2=6.4736')
pause(0.2)

disp('方法三：拉普拉斯变换')
L10=1.0;  L20=1.0;      %%初位移
syms  k  M  m  w  XL10  XL20  XL30  real      %%指定变量名
ddXL1=diff(sym('XL1(t)'),2);      %%表示 XL1 的二阶导数
dXL1=sym('diff(XL1(t),t)');      %%表示 XL1 的一阶导数
XL1=sym('XL1(t)');      %%表示位移 XL1

ddXL2=diff(sym('XL2(t)'),2);      %%表示 XL2 的二阶导数
dXL2=sym('diff(XL2(t),t)');      %%表示 XL2 的一阶导数
XL2=sym('XL2(t)');      %%表示位移 XL2

ddXL3=diff(sym('XL3(t)'),2);      %%表示 XL3 的二阶导数
dXL3=sym('diff(XL3(t),t)');      %%表示 XL3 的一阶导数
XL3=sym('XL3(t)');      %%表示位移 XL3

syms t s
eq1=m*ddXL1-k*(XL2-XL1);      %%表示方程一
eq2=M*ddXL2+k*(XL2-XL1)-k*(XL3-XL2);      %%表示方程二
eq3=m*ddXL3+k*(XL3-XL2);      %%表示方程三
```

```
L1=laplace(eq1,t,s);      %%对方程一进行拉普拉斯变换
L2=laplace(eq2,t,s);      %%对方程二进行拉普拉斯变换
L3=laplace(eq3,t,s);      %%对方程三进行拉普拉斯变换

syms LXL1 LXL2
NXL1=subs(L1,'XL1(0)','XL2(0)','XL3(0)','D(XL1)(0)','D(XL2)(0)',...
'D(XL3)(0)','XL10',0,0,0,0,0);      %%替换初值
NXL2=subs(L2,'XL1(0)','XL2(0)','XL3(0)','D(XL1)(0)','D(XL2)(0)',...
'D(XL3)(0)','XL10',0,0,0,0,0);      %%替换初值
NXL3=subs(L3,'XL1(0)','XL2(0)','XL3(0)','D(XL1)(0)','D(XL2)(0)',...
'D(XL3)(0)','XL10',0,0,0,0,0);      %%替换初值

NNXL1=subs(NXL1,'laplace(XL1(t),t,s)','laplace(XL2(t),t,s)',...
'laplace(XL3(t),t,s)','LL1','LL2','LL3');      %%替换拉氏符号
CXL1=collect(NNXL1,'LL1');      %%合并同类项
NNXL2=subs(NXL2,'laplace(XL1(t),t,s)','laplace(XL2(t),t,s)',...
'laplace(XL3(t),t,s)','LL1','LL2','LL3');      %%替换拉氏符号
CXL2=collect(NNXL2,'LL2');      %%合并同类项
NNXL3=subs(NXL3,'laplace(XL1(t),t,s)','laplace(XL2(t),t,s)',...
'laplace(XL3(t),t,s)','LL1','LL2','LL3');      %%替换拉氏符号
CXL3=collect(NNXL3,'LL3');      %%合并同类项

    %%解变换后的方程
[j1,j2,j3]=solve(CXL1,CXL2,CXL3,'LL1','LL2','LL3');
XL1=ilaplace(j1,s,t)      %%逆变换求位移 XL1
XL2=ilaplace(j2,s,t)      %%逆变换求位移 XL2
XL3=ilaplace(j3,s,t)      %%逆变换求位移 XL3

XL1=eval(subs(XL1,m,M,k,XL10,3,4,50,0.2));      %%求数值结果
XL2=eval(subs(XL2,m,M,k,XL10,3,4,50,0.2));      %%求数值结果
XL3=eval(subs(XL3,m,M,k,XL10,3,4,50,0.2));      %%求数值结果
XL1=vpa(XL1,4)      %%解数值化
XL2=vpa(XL2,4)      %%解数值化
XL3=vpa(XL3,4)      %%解数值化
XL1=inline(char(XL1),'t');      %%先把 XL1 变成字符串，再变成函数
XL2=inline(char(XL2),'t');
XL3=inline(char(XL3),'t');
```

```
pause(0.2)

L=1;      %%弹簧的长度
a=0.2;      %%小球直径，也是大球的半径
t=0:0.01:33;
XL1=XL1(t);      %%一般振动时三质点的运动方程
XL2=XL2(t);
XL3=XL3(t);
sp{1}=' 一般振动模拟 ';      %%标注文字
sp{2}=' 简正模 1';
sp{3}=' 简正模 2';
figure
axis([-1,2*L+4*a+1,-1,1]);
hold on

for n=1:3
   cla
   text(1,0.8,sp{n},'FontSize',16,'FontName',' 宋体 ',...
      'FontWeight','bold','Color',[ 0 0 0.6275])
   if n==1
   elseif n==2
      XL1=.4000e-1*cos(6.456*t);
      XL2=-.6000e-1*cos(6.456*t);
      XL3=.4000e-1*cos(6.456*t);
   else n==3
      XL1=.1000*cos(4.082*t);
      XL2=.6000e-1*ones(1,1000);
      XL3=-.1000*cos(4.082*t);
   end

   qiou1=line(0,0,'color','r','marker','.','markersize',50,...
   'erasemode','xor');      %%画球
   qiou2=line(L+3*a/2,0,'color','b','marker','.',...
   'markersize',80,'erasemode','xor');
   qiou3=line(2*L+3*a,0,'color','r ','marker','.',...
   'markersize',50,'erasemode','xor');
```

```
        xx1=linspace(a/2,L+a/2,10);      %%以下画弹簧
        xx2=linspace(L+5*a/2,2*L+5*a/2 ,10);
        yy1=[0,-0.04,0.04,-0.04,0.04,-0.04,0.04,-0.04,0.04,0];
        yy2=yy1;
        tanhuang1=line(xx1,yy1,'color','g','linestyle','-',...
        'erasemode','xor','linewidth',2);
        tanhuang2=line(xx2,yy2,'color','g','linestyle','-',...
        'erasemode','xor','linewidth',2);
        for i=1:1000       %%动画
           xx1=linspace(XL1(i)+a/2,L+a/2+XL2(i),10);
           xx2=linspace(XL2(i)+L+5*a/2,2*L+5*a/2+XL3(i),10);
           set(tanhuang1,'XData',xx1,'YData',yy1);
           set(tanhuang2,'XData',xx2,'YData',yy2);
           set(qiou1,'XData',XL1(i),'YData',0);
           set(qiou2,'XData',L+3*a/2+XL2(i),'YData',0);
           set(qiou3,'XData',2*L+3*a+XL3(i),'YData',0);
           drawnow;
        end
    end
    disp('the end')
```

函数文件是一个独立的文件，文件名为 szydxtfun.m, 其中 $y_1 = x_1$，$y_2 = x_2$，$y_3 = x_3$，$y_4 = \mathrm{d}x_1/\mathrm{d}t$，$y_5 = \mathrm{d}x_2/\mathrm{d}t$，$y_6 = \mathrm{d}x_3/\mathrm{d}t$ 。

```
function ydot=szydxtfun(t,y,flag)
m=3;
M=4;
k=50;
ydot=[y(4);
     y(5);
     y(6);
     k/m*(y(2)-y(1));
     k/M*(y(3)-y(2))-k/M*(y(2)-y(1));
     k/m*(y(2)-y(3))];
```

2.18 苯环模型

1. 实验题目

研究简化的苯分子模型的振动。苯分子可简化为图 2.32 所示的六个自由度的微振动系统模型，即六个质量均为 m 的相同的质点被约束在光滑的固定水平圆环上运动。珠子之间用相同的无质量的弹簧连接，弹簧的倔强系数为 k 。

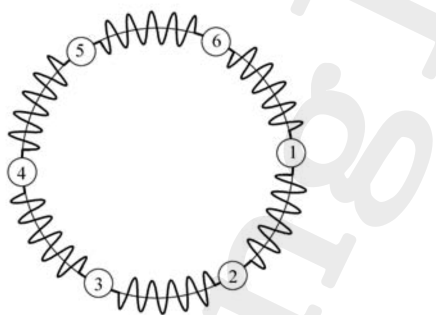


图 2.32　苯环的简化模型

2. 实验目的和要求

(1) 用矩阵方法求简正频率。可直接求解系统运动微分方程，亦可利用运动的对称性将自由度减少后列方程求解。本题可以看作是将 2.16 节的方法推广到自由度更多的系统，从中可以看到计算机求解的优点。

(2) 动画模拟六个简正模式和系统的一般运动。

(3) 学习用指令 diag 构造对角矩阵。

3. 解题分析

苯分子模型有六个自由度，以质点相对自身平衡位置移动的弧坐标 s_1，s_2，s_3，s_4，s_5，s_6 作为广义坐标。

方法一：用矩阵方法求简正频率

系统的动能为

$$
\begin{aligned}
T \;=\; & \frac{1}{2}m\left(\frac{\mathrm{d}s_1}{\mathrm{d}t}\right)^2+\frac{1}{2}m\left(\frac{\mathrm{d}s_2}{\mathrm{d}t}\right)^2+\frac{1}{2}m\left(\frac{\mathrm{d}s_3}{\mathrm{d}t}\right)^2+\\
& \frac{1}{2}m\left(\frac{\mathrm{d}s_4}{\mathrm{d}t}\right)^2+\frac{1}{2}m\left(\frac{\mathrm{d}s_5}{\mathrm{d}t}\right)^2+\frac{1}{2}m\left(\frac{\mathrm{d}s_6}{\mathrm{d}t}\right)^2
\end{aligned}
$$

以平衡位置为势能零点，系统的势能为：

$$
V \;=\; \frac{1}{2}k\left(s_2-s_1\right)^2+\frac{1}{2}k\left(s_3-s_2\right)^2+\frac{1}{2}k\left(s_4-s_3\right)^2+
$$

$$\frac{1}{2}k\left(s_5-s_4\right)^2+\frac{1}{2}k\left(s_6-s_5\right)^2+\frac{1}{2}k\left(s_1-s_6\right)^2$$

令 $u=2k-m\,\omega^2$, 写出系统的本征方程为

$$\begin{vmatrix} u & -k & 0 & 0 & 0 & 0 & -k \\ -k & u & -k & 0 & 0 & 0 & 0 \\ 0 & -k & u & -k & 0 & 0 & 0 \\ 0 & 0 & -k & u & -k & 0 & 0 \\ 0 & 0 & 0 & -k & u & -k & 0 \\ 0 & 0 & 0 & 0 & -k & u & -k \\ -k & 0 & 0 & 0 & 0 & -k & u \end{vmatrix}=0$$

求出简正频率

$$\omega_1=\sqrt{\frac{k}{m}},\quad \omega_2=\sqrt{\frac{3k}{m}},\quad \omega_3=\sqrt{\frac{k}{m}}$$

$$\omega_4=2\sqrt{\frac{k}{m}},\quad \omega_5=0,\quad \omega_6=\sqrt{\frac{3k}{m}}$$

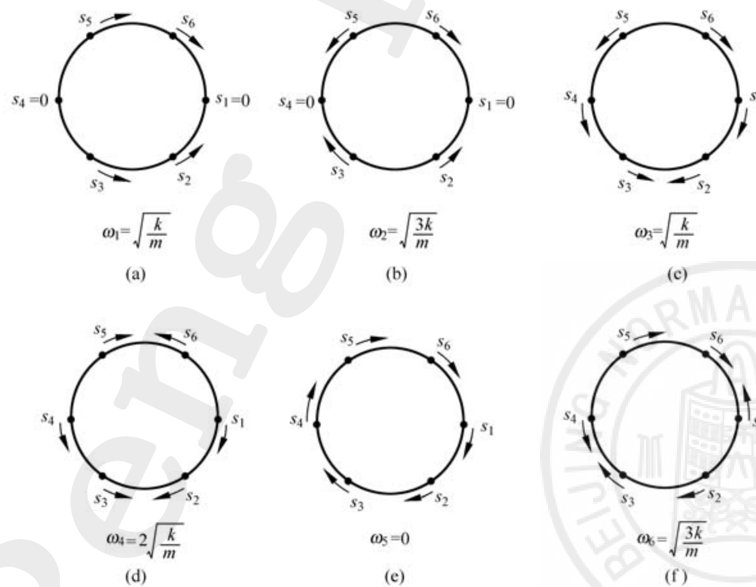


图 2.33 苯环的 6 种简正振动

简正频率中出现了 $\omega_1=\omega_3,\ \omega_2=\omega_6$，但不能认为简正模式减少。根据线性代数理论，说明对应同一个简正频率存在两个线性无关的本征矢（简正模式）。即 6 个简正频率对应于 6 个简正模式，其示意图如图 2.33 所示。

方法二：运用对称性求解

根据对称性，可直接分析得出苯环模型有六种如图 2.33 的简正模式。解题步骤是先将自由度减少，再用一般方法求解。求解过程如下：

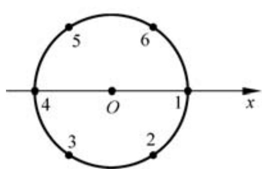


图 2.34　苯环振动对 x 轴的对称性

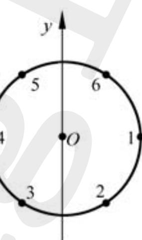


图 2.35　苯环振动对 y 轴的对称性

(1) 设 1, 4 质点静止，其他四个质点的运动相对于通过 1, 4 质点的 Ox 轴对称（见图 2.34），$s_2=s_6,\ s_3=s_5,\ s_1=0,\ s_4=0$，自由度由 6 减少到 2，广义坐标选为 $s_2,\ s_3$。

系统的动能和势能分别为

$$T=m\left(\frac{\mathrm{d}s_2}{\mathrm{d}t}\right)^2+m\left(\frac{\mathrm{d}s_3}{\mathrm{d}t}\right)^2$$

$$V=ks_2^2+k\left(s_3-s_2\right)^2+ks_3^2$$

含 ω^2 的行列式方程为

$$\begin{vmatrix} 2k-m\omega^2 & -k \\ -k & 2k-m\omega^2 \end{vmatrix}=0$$

得简正频率

$$\omega_1=\sqrt{\frac{k}{m}},\quad \omega_2=\sqrt{\frac{3k}{m}}$$

分别对应 ω_1,ω_2 的两个本征矢量为

$$A_1=\begin{bmatrix} A_{21} \\ A_{31} \end{bmatrix}=\begin{bmatrix} A_{21} \\ A_{21} \end{bmatrix},\quad A_2=\begin{bmatrix} A_{22} \\ A_{32} \end{bmatrix}=\begin{bmatrix} A_{22} \\ A_{22} \end{bmatrix}$$

简谐振动方程分别为

$$\left.\begin{aligned} s_2 &= A_{21}\cos(\omega_1 t+\varphi_1) \\ s_3 &= A_{21}\cos(\omega_1 t+\varphi_1) \end{aligned}\right\}$$

$$\left.\begin{aligned} s_1 &= A_{22}\cos(\omega_2 t+\varphi_2) \\ s_3 &= -A_{22}\cos(\omega_2 t+\varphi_2) \end{aligned}\right\}$$

与 ω_1 对应的简正模如图 2.33(a)，与 ω_2 对应的简正模如图 2.33(b)。

(2) 设系统各质点的运动对 Oy 轴对称，如图 2.35 所示，$s_2=s_3$，$s_1=s_4$，$s_6=s_5$，自由度变成 3。选择 s_2, s_1, s_6 为广义坐标。

系统的动能和势能分别为

$$T = m\left(\frac{\mathrm{d}s_1}{\mathrm{d}t}\right)^2 + m\left(\frac{\mathrm{d}s_2}{\mathrm{d}t}\right)^2 + m\left(\frac{\mathrm{d}s_6}{\mathrm{d}t}\right)^2$$

$$V = k(s_2-s_6)^2 + k(s_1-s_6)^2 + \frac{1}{2}k(2s_2)^2 + \frac{1}{2}k(2s_6)^2$$

利用下面的方程，求出简正频率：

$$\begin{vmatrix} 2k-m\omega^2 & -k & -k \\ -k & 3k-m\omega^2 & 0 \\ -k & 0 & 3k-m\omega^2 \end{vmatrix} = 0$$

$$\omega_3 = \sqrt{\frac{k}{m}},\quad \omega_3' = \sqrt{\frac{k}{m}},\quad \omega_4 = 2\sqrt{\frac{k}{m}}$$

对于 ω_3，本征矢为

$$A_3 = \begin{bmatrix} A_{13} \\ A_{23} \\ A_{63} \end{bmatrix} = \begin{bmatrix} A_{13} \\ \frac{1}{2}A_{13} \\ \frac{1}{2}A_{13} \end{bmatrix}$$

简正振动方程

$$\left.\begin{aligned} s_1 &= A_{13}\cos(\omega_3 t+\varphi_3) \\ s_2 &= \frac{1}{2}A_{13}\cos(\omega_3 t+\varphi_3) \\ s_6 &= \frac{1}{2}A_{13}\cos(\omega_3 t+\varphi_3) \end{aligned}\right\}$$

简正模式如图 2.33(c) 所示。

对于 ω_3'，经计算，其简正模式与图 2.33(b) 相同，没有给出新的模式。

对于 ω_4，本征矢为

$$A_4=\begin{bmatrix}A_{14}\\A_{24}\\A_{64}\end{bmatrix}=\begin{bmatrix}A_{14}\\-A_{14}\\-A_{14}\end{bmatrix}$$

简正振动方程为

$$\left.\begin{aligned}s_1&=A_{14}\cos\left(\omega_4 t+\varphi_4\right)\\s_2&=-A_{14}\cos\left(\omega_4 t+\varphi_4\right)\\s_6&=-A_{14}\cos\left(\omega_4 t+\varphi_4\right)\end{aligned}\right\}$$

简正模式如图 2.33(d) 所示。

(3) 根据力学系统对圆心 O 有中心反演对称性，设各质点位移方向相同，$s_1=s_4$, $s_2=s_5$, $s_3=s_6$，将系统自由度设计为 3，取 s_1、s_2、s_3 为广义坐标。

系统的动能和势能分别写成

$$T=m\left(\frac{\mathrm{d}s_1}{\mathrm{d}t}\right)^2+m\left(\frac{\mathrm{d}s_2}{\mathrm{d}t}\right)^2+m\left(\frac{\mathrm{d}s_3}{\mathrm{d}t}\right)^2$$

$$V=k\left(s_1-s_2\right)^2+k\left(s_2-s_3\right)^2+k\left(s_3-s_1\right)^2$$

可得

$$\begin{vmatrix}2k-m\omega^2&-k&-k\\-k&2k-m\omega^2&0\\-k&-k&2k-m\omega^2\end{vmatrix}=0$$

求出三个简正频率

$$\omega_5=0,\quad \omega_6=\omega_6'=\sqrt{\frac{3k}{m}}$$

与 ω_6 对应的运动是所有质点以相同速率绕 O 点作纯转动，如图 2.33(e) 所示。为了简单，在程序中取初始速度为零。

对 ω_6 和 ω_6'，为满足中心反演对称性，要求三个质点的运动满足两种模式。

第一种模式是

$$s_2=s_3,\quad s_1=-2s_2=-2s_3$$

第二种模式是

$$s_1 = 0, \quad s_2 = -s_3$$

前一种是新的简正模式，如图 2.33(f) 所示。后一种模式与图 2.33(b) 的模式相同。这两种模式都能使 $s_1 + s_2 + s_3 = 0$ 。这是质点运动所要求的。

在编程计算中，计算各质点位移的过程完全与 2.16 节弹簧连接的耦合摆相同，只是变量的个数增加了。这里不再叙述。为了构造对角矩阵，使用了指令 diag, 其用法在第一章中有过介绍。

在程序中，有一个子程序是用来画运动中的弹簧，其思路是，先以某个固定点 A 画一条正弦曲线来表示弹簧，然后以 A 点为轴转过角度 q_1, 最后再做一次坐标变换，把直的弹簧变换成沿圆环弯曲的弹簧。

4. 思考题

请找出动画中各种振动模式与解析解的各种圆频率的对应关系。

5. 参考程序

主程序的文件名是 bh.m 。

```
function bh
m=1; k=50;
    %%计算本征值与本征频率
S=m/k*diag(ones(1,6));
P=2*diag(ones(1,6))-diag(ones(1,5),-1)-diag(ones(1,5),+1);
P(1,6)=-1;    P(6,1)=-1;
[JM,JBB]=eig(S\P)
JB=abs(sqrt(JBB));
    %%给定位移表达式中的各项的系数
a1=[0.2, 0, 0, 0, 0, 0, 0.1];
a2=[ 0, 0.2, 0, 0, 0, 0, 0.1];
a3=[ 0, 0, 0.2, 0, 0, 0, 0.1];
a4=[ 0, 0, 0, 0.2, 0, 0, 0.1];
a5=[ 0, 0, 0, 0, 0.2, 0, 0.1];
a6=[ 0, 0, 0, 0, 0, 0.2, 0.1];
phi1=0; phi2=0; phi3=0;       %%设置初始位相
phi4=0; phi5=0; phi6=0 ;
t=0:0.01:4;       %%质点运动的时间
r=1;      %%圆环的半径
```

```
figure
for kk=1:7
axis([-1.5*r 1.5*r -1.5*r 1.5*r]);
axis equal
hold on
plot(r.*cos(0:0.1:2*pi),r.*sin(0:0.1:2*pi),'y')     %%画参考圆
   if kk==7     %%加文字标注
      title(' 一般模式 ')
   else
      ti1=' 简正模 \cdot\cdot\cdot'; ti2=int2str(kk);
      ti=[ti1,ti2];
      title(ti);
   end
   for i=1:6     %%计算各质点的中心位置
   ss=a1(kk)*JM(i,1)*cos(JB(1,1)*t+phi1)+a2(kk)*JM(i,2)...
    *cos(JB(2,2)*t+phi2)+a3(kk)*JM(i,3)*cos(JB(3,3)*t+phi3)...
    +a4(kk)*JM(i,4)*cos(JB(4,4)*t+phi4)+a5(kk)*JM(i,5)...
    *cos(JB(5,5)*t+phi5)+a6(kk)*JM(i,6)*cos(JB(6,6)*t+phi6);
   x{i}= r.*cos((i)*pi/3-ss./r);
   y{i}= r.*sin((i)*pi/3-ss./r);
end

   for i=1:5
      [xp,yp]=plotstring(x{i}(1),y{i}(1),x{i+1}(1),y{i+1}(1),r);
      h{i}=plot(xp,yp,'erasemode','xor','linewidth',1.5);    %%画弹簧
      hh{i}=plot(x{i}(1),y{i}(1),'erasemode','xor','marker','o',...
         'markersize',25,'linewidth',2.5,'color','r');     %%画小圆圈
      ii=7-i;
      hhh{i}=text(x{i}(1)-0.08,y{i}(1)+0.01,int2str(ii),...
         'fontsize',14,'erasemode','xor');     %%质点的编号
   end
   [xp6,yp6]=plotstring(x{6}(1),y{6}(1),...
      x{1}(1),y{1}(1),r);     %%画第 6 个质点的图像
   h6=plot(xp6,yp6,'erasemode','xor','linewidth',1.5);
   hh6=plot(x{6}(1),y{6}(1),'erasemode','xor','marker','o',...
    'markersize',25,'linewidth',2.5,'color','r');
```

```
    hhh6=text(x{6}(1)-0.08,y{6}(1)+0.01,'1','fontsize',...
    14,'erasemode','xor');
    for j=2:2:401      %%作动画
      for i=1:5
        [xp,yp]=plotstring(x{i}(j),y{i}(j),x{i+1}(j),y{i+1}(j),r);
        set(h{i},'xdata',xp,'ydata',yp);
        set(hh{i},'xdata',x{i}(j),'ydata',yi(j));
        set(hhh{i},'position',[x{i}(j),y{i}(j)]);
      end
    [xp6,yp6]=plotstring(x{6}(j),y{6}(j),x{1}(j),y{1}(j),r);
    set(h6,'xdata',xp6,'ydata',yp6);
    set(hh6,'xdata',x{6}(j),'ydata',y{6}(j));
    set(hhh6,'position',[x{6}(j),y{6}(j)]);
    drawnow;
end
cla      %%清除图形窗口
end
close(gcf);      %%关闭图形窗口

    %%画弹簧的变换子函数
function [xp,yp]=plotstring(xa,ya,xb,yb,r)
    %%以下各句是将弹簧的各个起点从质点的中心移到小圆圈的边上
[xa1,ya1]=cart2pol(xa,ya);
xa1=xa1+0.13;
[xa,ya]=pol2cart(xa1,ya1);
[xb1,yb1]=cart2pol(xb,yb);
xb1=xb1-0.13;
[xb,yb]=pol2cart(xb1,yb1);
a=0.13;  n=5;      %%弹簧的直径及圈数
q2=[ ];
d=sqrt((xa-xb)^2+(ya-yb)^2);      %%弹簧的长度
w=2*pi*n/d;
x=xa:0.02:(xa+d);      %%弹簧的 x 坐标
y=a.*sin(w.*(x-xa));      %%弹簧的 y 坐标
if xa>xb      %%计算弹簧转动的角度
```

```
   q1=pi+atan((ya-yb)/(xa-xb));
else
   q1=atan((ya-yb)/(xa-xb));
end
xd=xa+(x-xa).*cos(q1);      %%旋转弹簧
yd=ya+(x-xa).*sin(q1);

    %%以下是对弹簧的各点作变换
for i=1:length(y)
   if xd(i)<0
      q2a=pi+atan((yd(i))/(xd(i)));
      q2=[q2 q2a];
   else
      q2a=atan((yd(i))/(xd(i)));
      q2=[q2 q2a];
   end
end
xp=(r+y).*cos(q2);
yp=(r+y).*sin(q2);
```




2.19 自激振动

1. 实验题目

研究范 · 德 · 波耳 (Van der pol) 方程

$$\frac{\mathrm{d}^2x}{\mathrm{d}t^2}-\mu\left(x_0^2-x^2\right)\frac{\mathrm{d}x}{\mathrm{d}t}+\omega_0^2x=0 \tag{2.19.1}$$

所描述的非线性有阻尼的自激振动系统，其中 μ 是一个小的正的参量， x_0 是常数。下面简称范 · 德 · 波耳方程为 VDP 方程。

在 VDP 方程中，增加外驱动力 $V\cos\omega t$ 项所得到的方程

$$\frac{\mathrm{d}^2x}{\mathrm{d}t^2}-\mu\left(x_0^2-x^2\right)\frac{\mathrm{d}x}{\mathrm{d}t}+\omega_0^2x+V\cos\omega t=0 \tag{2.19.2}$$

称强迫 VDP 方程，其中外驱动力的振幅、角频率分别是 V 和 ω 。试研究强迫 VDP 方程的行为。

2. 实验目的和要求

(1) 演示 VDP 方程所描述的系统在非线性能源供给下，从任意初始条件出发都能产生稳定的周期性运动。

(2) 采用庞加莱映像，演示强迫 VDP 方程在不同参数下所存在四种吸引子，即周期 1 吸引子、周期 2 吸引子、不变环面吸引子和奇怪吸引子。

(3) 对于强迫 VDP 方程，在 V 和 ω 为定值条件下，逐渐增大 μ 值，将出现周期倍分岔和混沌现象。

3. 解题分析

自激系统是一个非线性有阻尼的振动系统，在运动过程中伴随有能量损耗。但系统存在一种机制，使能量能够由非振动的能源通过系统本身的反馈调节，及时适量地得到补充，从而产生一个稳定的不衰减的周期运动，这样的振动称为自激振动。

对 VDP 方程，可从机械振动角度理解， $-\mu\left(x_0^2-x^2\right)$ 是阻尼系数，它是变化的。如果 $|x|>|x_0|$ ，则阻尼系数为正，系统将受阻尼，能量将逐渐减少；但如果 $|x|<|x_0|$ ，则发生负阻尼，意味着不仅不消耗系统的能量，反而给系统提供能量。此系统能通过自动的反馈调节，使得在一个振动过程中，补充的能量正好等于消耗的能量，从而系统作稳定的周期振动。

取方程中的 ${x_0}^2=1,{\omega_0}^2=1,\mu=0.3$ (这些值可适当调整)。给出任一初始条件，通过计算机数值求解可以证明它的相轨道都将趋向于一条闭合曲线，这一条闭合曲线，称为极限环，极限环以外的相轨道向里盘旋，而极限环以内的相轨

道则向外盘旋，都趋向极限环 (如图 2.36 所示)，说明不论初始情况如何，系统最终都到达以极限环描述的周期性运动。由于这段程序较简单，我们没有专门编写。事实上，只要将下面编写的关于强迫 VDP 方程的程序中令 $V=0$，$\mu=0.3$ 再取不同的初始条件，就能看到这个现象。

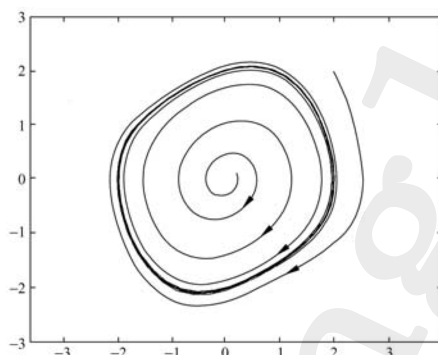


图 2.36　相图中极限环

下面研究强迫 VDP 方程的行为。我们同时采用时间历程图、相图和庞加莱映像图来研究系统在不同参数条件下的动力学行为，可以看到存在不同的吸引子，即周期 1 吸引子、周期 2 吸引子、不变环面吸引子和奇怪吸引子。

先对庞加莱映射作一简介。为了更清楚地了解运动的形态，庞加莱对连续运动的轨迹用一个截面 (叫庞加莱截面) 将其横截，那么根据轨迹在截面上穿过的情况，就可以简洁地判断运动的形态，由此所得图像叫庞加莱映像。在截面图上，轨迹下一次穿过截面的点 x_{n+1} 可以看成前一次穿过的点 x_n 的一种映射

$$x_{n+1}=f(x_n) \qquad (n=0,1,2,\cdots)$$

这个映射就叫庞加莱映射。它把一个连续的运动化为简洁的离散映射来研究。

在庞加莱映像中的不动点反映了相空间的周期运动。如果运动是二倍周期的，则庞加莱映像是两个不动点，四倍周期则有四个不动点等。

绘制庞加莱映像是在普通的相平面上进行，它不是像画相轨道那样随时间变化连续地画出相点，而是每隔一个外激励周期 $(T=2\pi/\omega)$ 取一个点。例如取样的时刻可以是 $t=0,T,2T\cdots$ 相应的相点记为 $P_0(x_0,y_0),P_1(x_1,y_1),P_2(x_2,y_2)\cdots$ 这些离散相点就构成了庞加莱映像。

设 $y_1=x$，$y_2=\mathrm{d}x/\mathrm{d}t$ ，则 (2.19.2) 式可化为

$$\left.\begin{aligned}\frac{\mathrm{d}y_1}{\mathrm{d}t}&=y_2\\ \frac{\mathrm{d}y_2}{\mathrm{d}t}&=\mu\left(x_0^2-y_1^2\right)y_2-\omega_0^2y_1-V\cos\omega t\end{aligned}\right\} \tag{2.19.3}$$

取 $x_0^2=1,\ \omega_0^2=1$，进行以下数值计算研究

(1) 在 $\mu=0.85,\ V=1,\ \omega=0.44$ 条件下，存在周期 1 吸引子，它的周期等于外激励的周期，代表主谐波运动。如图 2.37 所示。

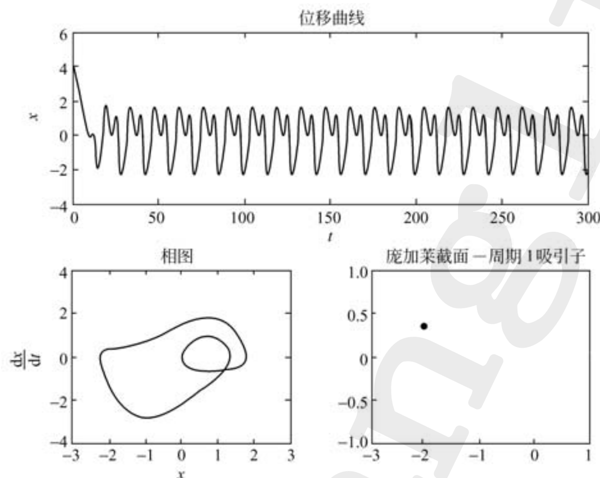


图 2.37 强迫 VDP 方程的振动的周期 1 吸引子

(2) 在 $\mu=1.02,\ V=1,\ \omega=0.44$ 条件下，存在周期 2 吸引子，它的周期等于外激励的整数倍，代表次谐波运动。如图 2.38 所示。

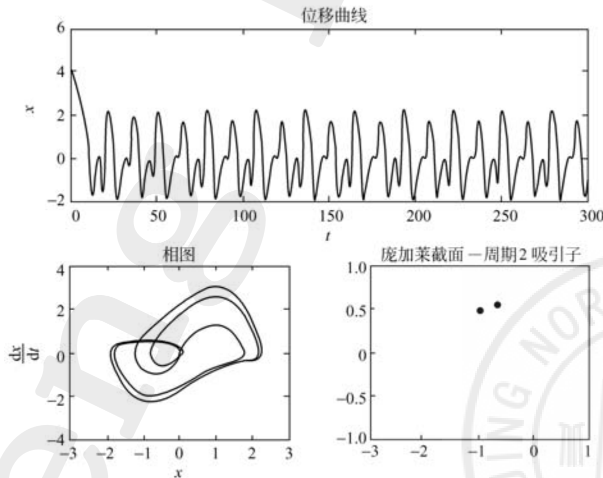


图 2.38 强迫 VDP 方程的振动的周期 2 吸引子

(3) 在 $\mu=0.66,\ V=1,\ \omega=0.44$ 条件下，存在不变环面吸引子，它代表准周

期（拟周期）运动。如图 2.39 所示。

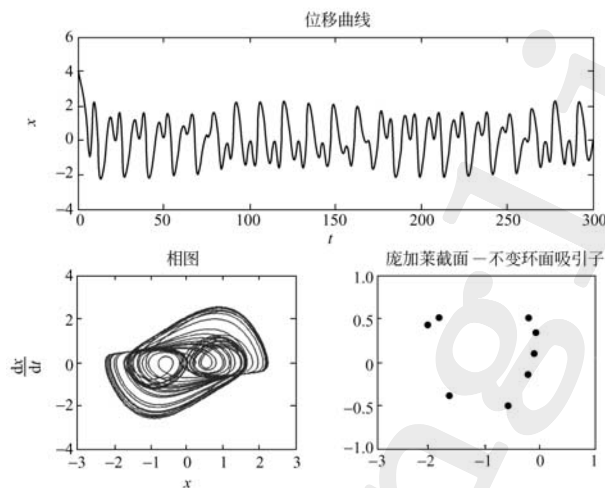


图 2.39 强迫 VDP 方程的振动的不变环面吸引子

(4) 在 $\mu=1.08$, $V=1$, $\omega=0.44$ 条件下，存在奇怪吸引子，代表混沌运动。如图 2.40 所示。

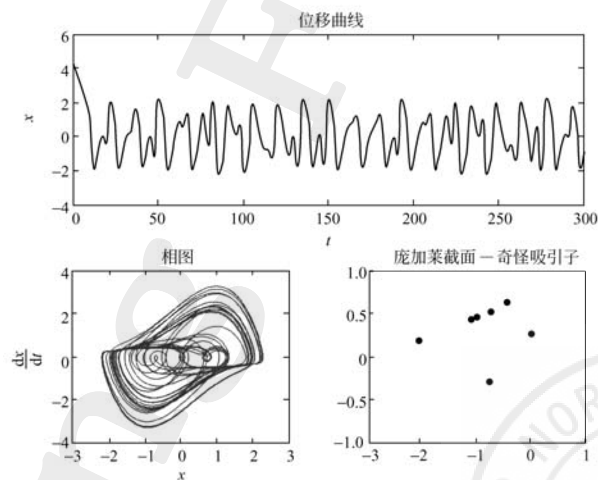


图 2.40 强迫 VDP 方程的振动的奇怪吸引子

(5) 保持 V 和 ω 为定值，逐渐增大 μ，将显示系统状态演化过程全貌的图，如图 2.41 所示。而前四种情况中，看到的只是 μ 取 4 个值的片段情况。图形显示，当 μ 由 0.9 连续变化到 1.2 时，系统运动状态逐渐由周期 1 过渡到周期 2（发

生了周期倍分岔）再过渡到混沌状态。

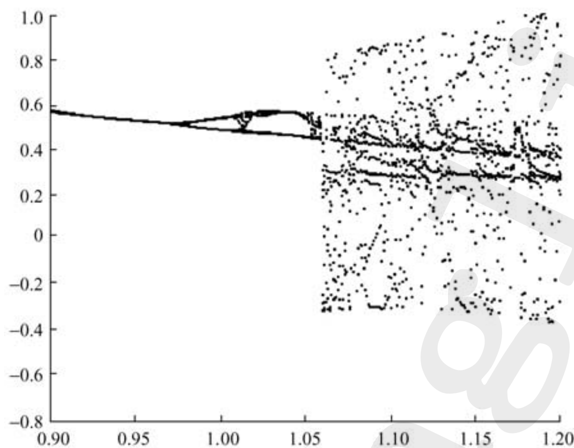


图 2.41 μ 值连续变化所产生的强迫 VDP 方程的庞加莱映像

在程序中，这几种过程的计算是相同的，所以用 for 循环来完成前面四种计算，这就是程序 zjzd.m 。计算中在每个外激励周期内计算 1000 个相点，为了作出庞加莱映像，每隔 1000 个点保留一个点数据，所以程序运行的时间较长。对第五种情况，由于计算量大，将它另外编写一个程序，这就是程序 zjzd1.m 。计算中在每个周期内计算 100 个相点，庞加莱映像是每隔 100 个点保留一个点数据。图 2.41 是在 CPU 为 P4 的计算机上运算约半小时所得到的结果。

4. 思考题

(1) 画出不同的吸引子的功率谱，观察它们的差别。

(2) 当 μ 值由 0.6 连续变化到 0.9 时，计算强迫 VDP 方程的庞加莱映像。

(3) 将参考程序 zjzd.m 中解微分方程的时间增加到足够长，在庞加莱映像图上可以看到一个更完整的奇怪吸引子形状。请试一试。

5. 参考程序

参考程序 zjzd.m 如下：

```
u=[0.85, 1.02, 0.66, 1.08];
x0=1;   w0=1;   v=1;   w=0.44;    T=2*pi/w;
str{1}=' 庞加莱截面—周期 1 吸引子 ';
str{2}=' 庞加莱截面—周期 2 吸引子 ';
str{3}=' 庞加莱截面—不变环面吸引子 ';
str{4}=' 庞加莱截面—奇怪吸引子 ';
```

```
for j=1:4
   [t,y]=ode23('zjzdfun',[0:T/1000:50*T],[4,4],[],u(j),x0,w0,v,w);

   figure
   subplot(2,1,1)
   plot(t,y(:,1));
   title(' 位移曲线 ');
   xlabel('t');ylabel('x');

   subplot(2,2,3)
   plot(y(3000:end,1),y(3000:end,2));
   axis([-3 3 -4 4])
   xlabel('x');ylabel('dx/dt');
   title(' 相图 ');

   subplot(2,2,4)
   axis([-3 1 -1 1])
   hold on
   for i=7000:1000:14000
      plot(y(i,1),y(i,2),'r.');
   end
   title(str{j});
end
```

参考程序 zjzd1.m 如下：

```
u=0.8:0.001:1.2;
v=1;  x0=1;  w0=1;  w=0.44;  T=2*pi/w;
axis([0.9 1.2 -0.8 1])
hold on
for j=1:length(u)
   [t,y]=ode23('zjzdfun',[0:T/100:70*T],[4,4],[ ],u(j),x0,w0,v,w);
   plot(u(j),y(500:100:1400,2),'linewidth',2);
end
```

函数文件是一个独立的文件，文件名为 zjzdfun.m 。

```
function ydot=zjzdfun(t,y,flag,u,x0,w0,v,w)
ydot=[y(2);
    u*(x0^2-y(1)^2)*y(2)-y(1)*w0^2-v*cos(w*t)];
```

2.20 倒摆的强迫振动

1. 实验题目

倒摆的实验装置如图 2.42 所示。倒摆是一个倒立的摆，其摆锤质量为 m，轻质杆长为 l，倒摆的底座以微小的幅度绕其中心作简谐摆动 $\varphi = A\cos\Omega t$ (A,Ω 为已知常数)，φ 角表示底座的垂线对铅垂线的偏离，θ 表示杆对底座的垂线的偏离。倒摆的势能曲线如图 2.43 所示。假设 $\varphi \ll \theta$，所以 θ 可近似表示杆对铅垂线的偏离；弹簧产生力矩为 $-c\theta$（c 为常量）；空气阻力为 $-\beta l(\frac{\mathrm{d}\theta}{\mathrm{d}t}+\frac{\mathrm{d}\varphi}{\mathrm{d}t}) \approx -\beta l\frac{\mathrm{d}\theta}{\mathrm{d}t}$（$\beta$ 为阻尼系数）。试研究倒摆在某些参数条件下的强迫振动。

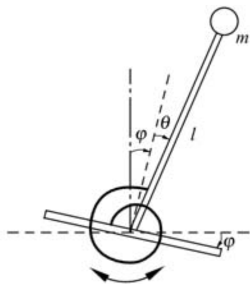


图 2.42 倒摆强迫振动的实验装置

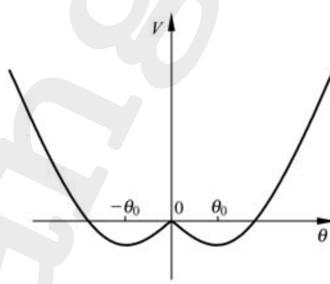


图 2.43 倒摆的势能曲线

2. 实验目的和要求

用数值计算证明在恰当的参数下，倒摆的强迫振动将出现混沌现象，为此从以下几方面考察混沌现象：

(1) 考察倒摆运动对初值的敏感依赖性，即在经过较长时间以后其运动是不可预测的。

(2) 通过相图的演变，观察奇怪吸引子的形成。

(3) 通过快速傅里叶变换，作出其功率谱，混沌现象的功率谱应为连续谱。

(4) 通过庞加莱截面，观察奇怪吸引子在此截面上的形状。

3. 解题分析

先导出倒摆强迫振动的运动微分方程：重力产生力矩为

$$mgl\sin(\theta+\varphi) \approx mgl(\theta - \frac{\theta^3}{6}) \tag{2.20.1}$$

根据质点的角动量定理，倒摆的运动微分方程为

$$ml^2\left(\frac{\mathrm{d}^2\theta}{\mathrm{d}t^2}+\frac{\mathrm{d}^2\varphi}{\mathrm{d}t^2}\right) = -c\theta + mgl\sin(\theta+\varphi) - \beta l^2\left(\frac{\mathrm{d}\theta}{\mathrm{d}t}+\frac{\mathrm{d}\varphi}{\mathrm{d}t}\right) \tag{2.20.2}$$

由于 $\frac{\mathrm{d}\varphi}{\mathrm{d}t} \ll \frac{\mathrm{d}\theta}{\mathrm{d}t}$, $\frac{\mathrm{d}^2\varphi}{\mathrm{d}t^2} \ll \frac{\mathrm{d}^2\theta}{\mathrm{d}t^2}$，所以上式可合理近似为

$$ml^2\frac{\mathrm{d}^2\theta}{\mathrm{d}t^2} + \beta l^2\frac{\mathrm{d}\theta}{\mathrm{d}t} + (c - mgl)\,\theta + \frac{1}{6}mgl\theta^3 = ml^2\Omega^2 A\cos\Omega t \tag{2.20.3}$$

我们仅在 $c < mgl$ 条件下进行研究。

为了对方程进行无量纲化，将方程 (2.20.3) 进行简化，并用大写的 T 表示时间，于是得到

$$\frac{\mathrm{d}^2\theta}{\mathrm{d}T^2} + \frac{\beta}{m}\frac{\mathrm{d}\theta}{\mathrm{d}T} - \frac{mgl - c}{ml^2}\theta + \frac{g}{6l}\theta^3 = A\Omega^2\cos\Omega T \tag{2.20.4}$$

设

$$\Omega_0^2 = \frac{mgl - c}{ml^2}$$

Ω_0 具有角频率的量纲。它的倒数给出问题中的一个时间尺度 T_0 。其次，在无驱动力时系统具有 3 个平衡位置： $\theta = 0$ 为不稳定的平衡位置；还有两个稳定平衡位置

$$\theta_0 = \pm\sqrt{6 - \frac{6c}{mgl}}$$

θ_0 的数值给出问题中角度的一个尺度（在一个问题中时间的参考尺度，空间的参考尺度往往不止一个）。现取 θ_0 作为角度的量度单位，取 $T_0(= 1/\Omega_0)$ 作为时间的量度单位，则无量纲的角度变量、无量纲的时间变量分别为

$$x = \frac{\theta}{\theta_0}, \qquad t = \frac{T}{T_0}$$

对 (2.20.4) 式进行变量变换，得

$$\frac{\mathrm{d}^2x}{\mathrm{d}t^2} + \frac{\beta}{m\Omega_0}\frac{\mathrm{d}x}{\mathrm{d}t} - x + x^3 = \frac{A}{\theta_0}\left(\frac{\Omega}{\Omega_0}\right)^2\cos\left(\frac{\Omega}{\Omega_0}t\right)$$

现分别引入无量纲的阻尼系数、无量纲的角频率和无量纲的驱动力的振幅

$$\delta = \frac{\beta}{m\Omega_0}, \qquad \omega = \frac{\Omega}{\Omega_0}, \qquad f = \frac{A}{\theta_0}\left(\frac{\Omega}{\Omega_0}\right)^2$$

于是可得无量纲方程

$$\frac{\mathrm{d}^2x}{\mathrm{d}t^2} + \delta\frac{\mathrm{d}x}{\mathrm{d}t} - x + x^3 = f\cos\omega t \tag{2.20.5}$$

这就是著名的强迫杜芬 (Duffing) 振动方程。

无量纲化的好处有两方面：方程涉及的只是数量关系，变量可以代表不同学科领域中的量，尤其在运算时无需顾及单位换算，这对数值计算中初值选取是方便的；更重要的是取不同的长度单位和时间单位时（即研究问题的时空尺

度的选取），无量纲化后方程中各项系数的量级大小不同，如选取得合适，能使非线性项系数为小量，说明在这样的时空尺度内非线性的作用是弱的。在不同时空尺度研究同一方程，会显示出不同景象。

设 $x = y_1$, $\mathrm{d}x/\mathrm{d}t = y_2$ ，则方程 (2.20.5) 化为两个一阶方程

$$\left.\begin{aligned} \frac{\mathrm{d}y_1}{\mathrm{d}t} &= y_2 \\ \frac{\mathrm{d}y_2}{\mathrm{d}t} &= -\delta y_2 + y_1 - y_1^3 + f\cos\omega t \end{aligned}\right\}$$

进行数值计算时可选取参数 $\delta = 0.26$, $f = 2$, $\omega = 2$ 。

程序通过选择恰当的参数，对初始条件只有极微小差别的两种情况，画出的位移曲线如图 2.44 所示。我们看到在时间不长的初始阶段，两曲线的差别并不大，只是在长时间后两者才有明显差别，即在不长时间内它们的行为是可预言的，只有在长时间以后才变成无法预言的“随机”行为。这种由确定性方程产生的对初值敏感的现象通常称为混沌现象。

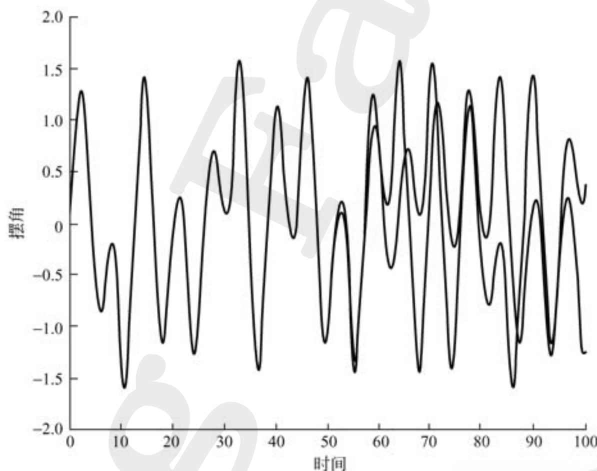


图 2.44 初始速度有微小差别的两条位移曲线

此系统具有双稳形态的势能曲线（见图 2.43），对于无量纲方程，不考虑外激励时，质点有两个稳定平衡位置 $(x = \pm 1)$ 和一个不稳定平衡位置 $(x = 0)$，说明质点可能在两个势阱中围绕各自的平衡位置振动。当振动受激励后质点运动可能到达不稳定平衡位置，此时将出现两种可能：一种是质点能够从一个势阱翻越势垒进入另一个势阱，一种是回到原来的势阱。因此质点运动的相图大体是这样：围绕一个稳定平衡位置振动几次后，跳到另一边，转而围绕另一稳

定平衡位置振动若干次后，又跳回这一边，围绕先前的平衡位置振动，…… 这样往复不止，而且每次情况都不相同。它不是周期性运动，相轨道极其复杂。它的轨道具有局部不稳定性，又具有全局稳定性，最终被吸引于相图中的某一位置附近，这种混沌的形态称为奇怪吸引子。可利用庞加莱映像显示强迫杜芬方程产生的奇怪吸引子。

由于混沌运动之非周期性、复杂的运动，它的功率谱不同于周期运动或准周期运动的离散功率谱，是连续谱。

程序中所使用的方法和语句，都是前面用过的，这里不再重复。为了得到庞加莱截面，需要在每个周期取一个点。所以在循环中，将步长增大了。程序运行所产生的一个图形如图 2.45 所示。

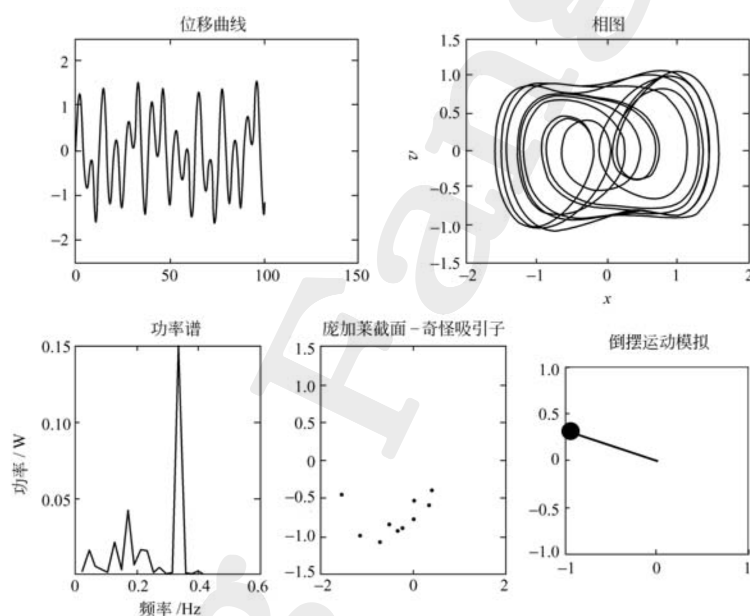


图 2.45　强迫杜芬方程产生的奇怪吸引子

4. 思考题

仿照 2.19 节的做法，取 $f=1$, $\omega=1$，令 δ 从 0.5 变化到 0.75，画出杜芬方程的庞加莱映像。

5. 参考程序

主程序的文件名是 db.m

```
%%设 v0 有微小的变化，比较解的变化情况
```

```
x0=0.1;v0=0.1;
[t,u]=ode45('dbfun',[0:0.01:100],[x0,v0],[],0.78);
[t1,u1]=ode45('dbfun',[0:0.01:100],[x0,v0-0.001],[],0.78);
figure
plot(t,u(:,1),'r',t1,u1(:,1),'g')
xlabel('时间');     ylabel('摆角');
title('混沌状态下初条件有微小差异会形成的两条分开的曲线');
```

%%当 d=2，为周期 1 吸引子；当 d=0.98 为周期 2 吸引子；当 d=0.78 为奇怪吸引子，读者可以改变 d 值，以观察不同的情况

```
d=[2,0.98,0.78];
str{1}='庞加莱截面—周期 1 吸引子';
str{2}='庞加莱截面—周期 2 吸引子';
str{3}='庞加莱截面—奇怪吸引子';
for j=1:3
   [t,u]=ode45('dbfun',[0:2*pi/300:100],[x0,v0],[],d(j));
   figure     %%适当放大图形窗口
   set(gcf,'unit','normalized','Position',[0.04 0.04 0.94 0.8]);

   subplot(2,2,1)      %%位移曲线
   plot(t,u(:,1))
   title('位移曲线');
   axis([0,150,-2.5,2.5]);

   subplot(2,2,2)      %%相图（奇怪吸引子）
   plot(u(:,1),u(:,2))
   title('相图');
   axis([-2 2 -1.5 1.5])
   xlabel('x');ylabel('v');

   Y=fft(u(:,1));      %%傅里叶功率分析
   Y(1)=[ ];
   n=length(Y);
   power=abs(Y(1:n/2)).^2/n^2;      %%功率
   freq=100*(1:n/2)./n;      %%频率
   subplot(2,3,4)
   plot(freq,power)
```

```
    axis([0 0.6 0 0.15])
    title(' 功率谱 ');
    xlabel(' 频率 /Hz');ylabel(' 功率 /w');

    subplot(2,3,5)      %%庞加莱截面
    plot(u(2000:300:4700,1),u(2000:300:4700,2),'r.');
    axis([-2 2 -1.5 1.5])
    title(str{j});

    subplot(2,3,6)     %%实物模拟图
    L=1;
    axis([-L L -L L])
    axis square
    title(' 倒摆运动模拟 ');
    hold on
    plot(0,0,'r.')
    ball=line(L*sin(x0),L*cos(x0),'color','r','marker','.',...
        'markersize',40,'erasemode','xor');
    gan=line([0,L*sin(x0)],[0,L*cos(x0)],'color','b','linewidth',2,...
       'erasemode','xor');

    for i=1:2:4770
        set(ball,'xData',L*sin(u(i,1)),'yData',L*cos(u(i,1)))
        set(gan,'xData',[0,L*sin(u(i,1))],'yData',[0,L*cos(u(i,1))])
        drawnow
    end
end
```

函数文件是一个独立的文件，文件名为 dbfun.m 。

```
function ydot=dbfun(t,y,flag,d)
r=1;w=1;
ydot=[y(2);
     -y(1)^3+y(1)-d*y(2)+r*cos(w*t)];
```

2.21 圆环的滚动

1. 实验题目

研究圆环在粗糙水平面上的滚动。设质量为 m，半径为 R 的匀质圆环，被限制在某一竖直平面内沿摩擦系数为 μ 的水平面滚动。开始时，用手按其后侧边缘，使圆环的质心获得一定的初速度 $\boldsymbol{v}_c$，同时环有向后转动的角速度 $\boldsymbol{\omega}_0$。

2. 实验目的和要求

分析并用动画模拟圆环在不同初始条件下的运动。

3. 解题分析

静止坐标系的 Ox 轴和圆盘转角 φ 的正方向如图 2.46 所示，坐标原点 O 为初始时刻圆环与水平面的接触点。圆环的运动可分成两个阶段讨论。

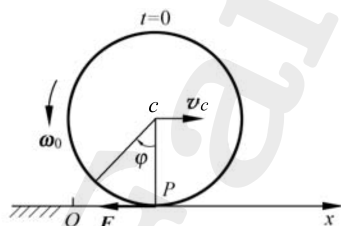


图 2.46 圆环的滚动

(1) 有滑滚动阶段

圆环除受到重力和桌面支持力作用外，还受到与质心速度方向相反的滑动摩擦力 $\boldsymbol{F}$ 的作用，$F=\mu mg$。接触点 P 的速度向前，小环作有滑滚动。根据刚体运动的质心运动定理和相对质心的角动量定理，圆环的运动微分方程为

$$\left.\begin{aligned}m\frac{\mathrm{d}^2x_c}{\mathrm{d}t^2}&=-\mu mg\\ mR^2\frac{\mathrm{d}^2\varphi}{\mathrm{d}t^2}&=-\mu mgR\end{aligned}\right\}\tag{2.21.1}$$

因为摩擦力 $\boldsymbol{F}$ 的方向与质心运动方向相反，摩擦力对质心的力矩的方向与圆环自转方向相反，所以质心速度和转动角速度都要减小。当运动到某一时刻，接触点 P 的速度为零，即 $v_p=v_c-R\mathrm{d}\varphi/\mathrm{d}t=0$，圆环进入无滑滚动阶段，方程 (2.21.1) 式不再适用。

当圆环达到无滑滚动时，上面两个方程不再适用。这就牵涉到判断积分何时终止的问题。本题用指令 events 来控制。当 $v_{cx}=R\mathrm{d}\varphi/\mathrm{d}t$ 时停止解方程。

(2) 无滑滚动阶段

该阶段共有三种可能情况：

第一种情况是圆环到达无滑时，刚好停止运动；

第二种情况是圆环达到无滑后，质心速度向后，圆环向后作无滑滚动；

第三种情况是圆环达到无滑后，质心继续向前运动，圆环改变自转方向向前作无滑滚动。

圆环运动中出现哪一种情况，取决于初始条件的选取。具体分析如下：

由 (2.21.1) 式，可解得

$$\left.\begin{aligned} v_{cx} &= v_0 - \mu g t \\ \frac{\mathrm{d}\varphi}{\mathrm{d}t} &= \omega_0 - \frac{\mu g t}{R} \end{aligned}\right\} \tag{2.21.2}$$

当 $v_{cx} = R\dfrac{\mathrm{d}\varphi}{\mathrm{d}t}$ 时，圆环进入无滑滚动阶段，由初始至刚达到无滑所需时间为

$$t = \frac{v_0 + \omega_0 R}{2\mu g} \tag{2.21.3}$$

在该时刻

$$\left.\begin{aligned} v_{cx} &= \frac{v_0 - \omega_0 R}{2} \\ \frac{\mathrm{d}\varphi}{\mathrm{d}t} &= \frac{\omega_0 R - v_0}{2R} \end{aligned}\right\} \tag{2.21.4}$$

经计算，圆环在无滑滚动阶段质心将作匀速直线运动，圆环的角速度 $\mathrm{d}\varphi/\mathrm{d}t$ 也保持不变。由 (2.21.4) 式得知三种可能运动对初始条件的要求是

若 $v_0 = \omega_0 R$，则 $v_{cx} = 0,\ \mathrm{d}\varphi/\mathrm{d}t = 0$，圆环达到无滑条件时停止运动；

若 $v_0 < \omega_0 R$，则 $v_{cx} < 0,\ \mathrm{d}\varphi/\mathrm{d}t > 0$，圆环达到无滑条件后向后作无滑滚动；

若 $v_0 > \omega_0 R$，则 $v_{cx} > 0,\ \mathrm{d}\varphi/\mathrm{d}t < 0$，圆环达到无滑滚动后向前作无滑滚动。

在数值计算时，也是分成两个阶段，首先考虑有滑滚动，设 $y_1 = x_c,\ y_2 = \mathrm{d}x_c/\mathrm{d}t,\ y_3 = \varphi,\ y_4 = \mathrm{d}\varphi/\mathrm{d}t$，则方程可写成

$$\left.\begin{aligned} \frac{\mathrm{d}y_1}{\mathrm{d}t} &= y_2 \\ \frac{\mathrm{d}y_2}{\mathrm{d}t} &= -\mu g \\ \frac{\mathrm{d}y_3}{\mathrm{d}t} &= y_4 \\ \frac{\mathrm{d}y_4}{\mathrm{d}t} &= -\frac{\mu g}{R} \end{aligned}\right\} \tag{2.21.5}$$

然后解这个微分方程，当 $v_0 = \omega_0 R$ 时，停止求解过程。从此以后，圆环以此刻的质心速度和角速度作无滑滚动。所以在程序中，适当增加了一段时间来观察此后的运动。再将这些数据做成动画即可。

4. 思考题

(1) 圆环作无滑滚动的质心速度和角速度是如何确定的？

(2) 用乒乓球代替圆环（二者的转动惯量不同），重做此题。

5. 参考程序

主程序的文件名是 yhgd.m 。

```
w0=[8 2 12];
st{1}='v_0=R\omega_0 时，圆环停止运动 ';        %%文字标注
st{2}='v_0>R\omega_0 时，圆环向前做无滑滚动 ';
st{3}='v_0<R\omega_0 时，圆环向后做无滑滚动 ';

for I=1:3
   x0=0; f0=0; v0=8; R=1;
   options=odeset('events','on');       %%启动事件判断功能
   [t,y,event]=ode23('yhgdfun',[0:0.05:100],[x0,v0,f0,w0(I)],options);
        %%达到无滑滚动后，将按无滑滚动的方程运动
   T=0:0.05:10;      %%设定无滑滚动的时间
   vx=[y(:,1);y(end,1)+y(end,2)*T'];
        %%有滑滚动及无滑滚动总的水平位移
   jd=[y(:,3);y(end,3)+y(end,4)*T'];
        %%有滑滚动及无滑滚动总的角位移

   axis([-10 30 -5 5])
   axis equal;
   hold on
   axis off
   set(gca,'position',[0.13 0 0.775 0.815 ]);      %%设置当前轴的范围

   set(gcf,'color',[0.4 0.8 0.96]);      %%设置背景颜色
   title(st{I},'fontsize',18,'fontname',' 隶书 ','color','r');
   theta=y(:,3); h=-2.5; r=2.7;

   xdian=vx+r*sin(jd); ydian=1-r*cos(jd);      %%变换为直角坐标
   dimian=line([-30 40],[h h],'linewidth',5,'color','k');      %%地面
   qiu=line(0,1,'marker','o','markersize',50,'linewidth',5,...
      'color','m','erasemode','xor');
   dian=line(0+r*sin(0),1-r*cos(0),'marker','.','markersize',5,...
```

```
        'color','b','erasemode','xor');
    gan=line([0,r*sin(0)],[1,1-r*cos(0)],'linestyle','-',...
        'linewidth',5, 'color','r','erasemode','xor');
    L=[10 100 200];    n=length(t);
    for i=1:n+L(I)
        set(qiu,'xdata',vx(i),'ydata',1);
        set(dian,'xdata',xdian(i),'ydata',ydian(i));
        set(gan,'xdata',[vx(i),xdian(i)],'ydata',[1,ydian(i)]);
        drawnow
        pause(0.1)
    end
    cla
end
```

函数文件是一个独立的文件，文件名为 yhgdfun.m 。

```
function varargout=yhgdfun(t,y,flag)
switch flag
case ''
   varargout{1}=f(t,y);
case 'events'
   [varargout{1:3}]=events(t,y);
otherwise
   error(['unknown flag' ''flag'' '.']);
end

function ydot=f(t,y)
R=1;
ydot=[y(2);
     -0.2*9.8;
      y(4);
     -0.2*9.8/R];

function [value,isterminal,direction]=events(t,y)
R=1;
value=y(2)+R*y(4);
isterminal=1;
direction=0;
```

2.22 惯量椭球

1. 实验题目

研究长方体刚体的惯量椭球。

2. 实验目的和要求

(1) 画出匀质长方体刚体对其几何中心点 B 的惯量椭球和对其一个顶点 A 的惯量椭球。切去椭球的 1/8，以便观察三个惯量主轴的方向；改变长方体边长，观察椭球的变化。

(2) 求出长方体以角速度 $\boldsymbol{\omega}$ 绕中心点转动时，长方体对瞬时轴的转动惯量和角动量 $\boldsymbol{L}$。

(3) 学习计算二重积分，学习椭球和剖面图的画法。

3. 题目分析

刚体作定点运动，以固定点 O 为原点，建立与刚体固连的坐标系 $Oxyz$。对 O 点的惯量椭球的方程是

$$I_{xx}x^2 + I_{yy}y^2 + I_{zz}z^2 + 2I_{yx}xy + 2I_{yz}yz + 2I_{zx}zx = 1$$

其中 I_{xx}, I_{yy}, I_{zz} 为转动惯量， I_{xy}, I_{yz}, I_{zx} 是三个惯量积。

$$I_{xx} = \iiint (y^2 + z^2)\mathrm{d}m$$

$$I_{yy} = \iiint (x^2 + z^2)\mathrm{d}m$$

$$I_{zz} = \iiint (x^2 + y^2)\mathrm{d}m$$

$$I_{xy} = I_{yx} = \iiint xy\mathrm{d}m$$

$$I_{yz} = I_{zy} = \iiint yz\mathrm{d}m$$

$$I_{zx} = I_{xz} = \iiint zx\mathrm{d}m$$

式中积分为遍及整个刚体的体积分。对于不同固定点，转动惯量和惯量积不同，所以惯量椭球形状各不相同。

图 2.47 和图 2.48 分别是以长方体某一顶点 A 和几何中心 B 为固定点的惯量椭球。下面计算图 2.47 所示的惯量椭球，先求出 A 点的惯量主轴方向。

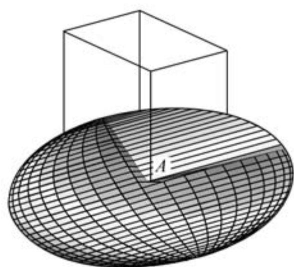


图 2.47 非主轴坐标系下的惯量椭球

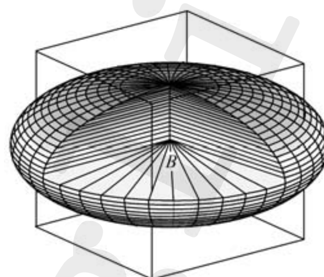


图 2.48 主轴坐标系下的惯量椭球

建立坐标系 $Ax_0y_0z_0$ ，各坐标轴分别沿长方体的三条边。该坐标系不是惯量主轴坐标系，其惯量张量为

$$I=\begin{bmatrix} I_{xx} & -I_{xy} & -I_{xz} \\ -I_{xy} & I_{yy} & -I_{yz} \\ -I_{xz} & -I_{yz} & I_{zz} \end{bmatrix}$$

使惯量张量对角化的本征方程是

$$\begin{vmatrix} I_{xx}-\lambda & -I_{xy} & -I_{xz} \\ -I_{xy} & I_{yy}-\lambda & -I_{yz} \\ -I_{xz} & -I_{yz} & I_{zz}-\lambda \end{vmatrix}=0$$

展开后可求得本征值 $\lambda_1, \lambda_2, \lambda_3$ 。又知如果角动量 $\boldsymbol{L}$ 沿主轴方向，即本征矢量的方向，则必然满足下列方程组：

$$\left.\begin{aligned} (I_{xx}-\lambda)\omega_x-I_{xy}\omega_y-I_{xz}\omega_z=0 \\ -I_{xy}\omega_x+(I_{yy}-\lambda)\omega_y-I_{yz}\omega_z=0 \\ -I_{xz}\omega_x-I_{yz}\omega_y+(I_{zz}-\lambda)\omega_z=0 \end{aligned}\right\}$$

将 λ_1 的值代入上式，可求出本征矢量 $\boldsymbol{\omega}_1$ 的方向。

设 ω_{1z} 为不为零的任意值，令

$$\frac{\omega_{1x}}{\omega_{1z}}=\alpha,\quad \frac{\omega_{1y}}{\omega_{1z}}=\beta$$

归一化条件为

$$\omega_{1x}^2+\omega_{1y}^2+\omega_{1z}^2=1$$

可知

$$\alpha\omega_{1z}^2+\beta\omega_{1z}^2+\omega_{1z}^2=1$$

得

$$\omega_{1z}=\frac{1}{(1+\alpha^2+\beta^2)^{1/2}}$$

于是 $\boldsymbol{\omega}_1$ 的方向余弦（主轴坐标系的主轴的方向余弦）为

$$\left.\begin{aligned}\alpha_{11}=\omega_{1x}&=\frac{\alpha}{(1+\alpha^2+\beta^2)^{1/2}}\\ \alpha_{12}=\omega_{1y}&=\frac{\beta}{(1+\alpha^2+\beta^2)^{1/2}}\\ \alpha_{13}=\omega_{1z}&=\frac{1}{(1+\alpha^2+\beta^2)^{1/2}}\end{aligned}\right\}$$

同样可求出本征矢量 $\boldsymbol{\omega}_2$ 和 $\boldsymbol{\omega}_3$ 的方向余弦分别是 $(\alpha_{21},\ \alpha_{22},\ \alpha_{23})$ 和（$\alpha_{31},\ \alpha_{32},\ \alpha_{33}$）。

三个本征矢量的方向分别是惯量主轴 x 轴、y 轴和 z 轴的方向。三个本征值 $\lambda_1,\ \lambda_2,\ \lambda_3$ 就是长方体对三根主轴的转动惯量。在 A 的主轴坐标系中，惯量椭球的方程为

$$\lambda_1x^2+\lambda_2y^2+\lambda_3z^2=1$$

现通过坐标变换矩阵，将非惯量主轴中的椭球的坐标变换为惯量主轴中的椭球坐标

$$\begin{bmatrix}x\\ y\\ z\end{bmatrix}=\begin{bmatrix}\alpha_{11}&\alpha_{12}&\alpha_{13}\\ \alpha_{21}&\alpha_{22}&\alpha_{23}\\ \alpha_{31}&\alpha_{32}&\alpha_{33}\end{bmatrix}\begin{bmatrix}x_0\\ y_0\\ z_0\end{bmatrix}$$

惯量椭球的引入有利于用几何方法研究刚体的定点运动，主要反映在以下三个方面。

(1) 由惯量椭球，可直接得到固定点 O 的惯量主轴，它们是惯量椭球的三个对称轴。若以惯量主轴作为动坐标系的坐标轴，椭球方程变成

$$I_{xx}x^2+I_{yy}y^2+I_{zz}z^2=1$$

但不论在哪个坐标系中，椭球的形状不会发生变化。

(2) 借助惯量椭球，可得到相对任何方向的瞬时转轴的转动惯量 I_l，$I_l=1/R^2$，R 是瞬时轴与椭球球面的交点到定点 O 的距离。

(3) 借助惯量椭球，可确定刚体以角速度 $\boldsymbol{\omega}$ 转动时的角动量 $\boldsymbol{L}$ 的方向和大小。 $\boldsymbol{L}$ 指向瞬时轴与椭球面交点的外法线方向。

已知角速度 $\boldsymbol{\omega}$ 的大小和方向，又知相对此方向的转动惯量，则角动量 $\boldsymbol{L}$ 在 $\boldsymbol{\omega}$ 方向上的投影为

$$L_\omega = \boldsymbol{L}\frac{\boldsymbol{\omega}}{\omega} = \boldsymbol{\omega}\cdot\boldsymbol{I}\cdot\frac{\boldsymbol{\omega}}{\omega} = I_\omega \omega$$

式中的 I_ω 是刚体对转轴的转动惯量。由于 $\boldsymbol{\omega}$ 、 $\boldsymbol{L}$ 的方向和 L_ω 都是已知的，因此可以得到 $\boldsymbol{L}$ 的大小。

在程序中，画椭球是利用球坐标系。画剖面图的方法是将不用的数值取为零。其余则基本上是按照公式计算。程序运行的结果如图 2.47 和图 2.48 所示。

4. 思考题

从物理意义上分析，所画的惯量椭球应该是什么形状，并与程序中画出的椭球相对照，判断你的分析是否正确。

5. 参考程序

主程序的文件名是 gltq.m 。

```
a=7;  b=8;  c=12;       %%正方形的边长
f1=inline('x.*x+y.*y','x','y');       %%两个被积函数
f2=inline('x.*y','x','y');

figure(1)     %%以 B 为固定点的主轴坐标系下的惯量椭球
set(gcf,'unit','normalized','position',[0.4 0.3 0.6 0.6]);
hold on
title(' 主轴坐标系下的惯量椭球 ');
axis off
view(-43,20)

    %%画立方体
cubx1=[0,a,a,a,a,0,0,0,0]-a/2;   cubx2=[0,0,a,a,a,a,0,0,0]-a/2;
cuby1=[0,0,0,b,b,b,b,0,0]-b/2;   cuby2=[0,0,0,0,b,b,b,b,0]-b/2;
cubz1=[c,c,0,0,c,c,0,0,c]-c/2;   cubz2=[c,0,0,c,c,0,0,c,c]-c/2;
plot3(2*cubx1,2*cuby1,2*cubz1,'b',2*cubx2,2*cuby2,2*cubz2,'b');

Ix=0.001*a*dblquad(f1,-b/2,b/2,-c/2,c/2)      %%计算惯量张量
Iy=0.001*b*dblquad(f1,-a/2,a/2,-c/2,c/2)
Iz=0.001*c*dblquad(f1,-a/2,a/2,-b/2,b/2)
```

```
n=40;     %%画椭球的数据
phi=(-n:2:n)/n*pi;  theta=(-n:2:n)'/n*pi/2;
xm=Ix*cos(theta)*cos(phi);
ym=Iy*cos(theta)*sin(phi);
zm=Iz*sin(theta)*ones(1,n+1);

for i=2:41      %%将椭球的一些范围的值设为 0 形成剖面图
   if theta(i)>0&theta(i)<pi/2
      ym(i,2:11)=0;     xm(i,2:11)=0;
   end
end
surf(xm,ym,zm);

figure(2)     %%以 A 为固定点的一般坐标系下的惯量椭球
set(gcf,'unit','normalized','position',[-0.1 -0.1 0.6 0.6]);
title(' 一般坐标系下的惯量椭球 ')
axis off
hold on
view(-30,42)

    %%画立方体
cubx1=[0,a,a,a,a,0,0,0,0];   cubx2=[0,0,a,a,a,a,0,0,0];
cuby1=[0,0,0,b,b,b,b,0,0];   cuby2=[0,0,0,0,b,b,b,b,0];
cubz1=[c,c,0,0,c,c,0,0,c];   cubz2=[c,0,0,c,c,0,0,c,c];
plot3(2*cubx1,2*cuby1,2*cubz1,'b',2*cubx2,2*cuby2,2*cubz2,'b');

    %%以下是画椭球
I(1,1) = 0.0004*a*dblquad(f1,0,b,0,c);      %%计算惯量张量各个分量
I(2,2) = 0.0004*b*dblquad(f1,0,a,0,c);
I(3,3) = 0.0004*c*dblquad(f1,0,a,0,b);
I(1,2) = 0.0004*-c*dblquad(f2,0,a,0,b);
I(1,3) = -b*0.0004*dblquad(f2,0,a,0,c);
I(2,3) = -a*0.0004*dblquad(f2,0,b,0,c);
I(2,1) = I(1,2);  I(3,1) = I(1,3);  I(3,2) = I(2,3);

o=0.35*pi;   p=0.4*pi;      %% 与坐标轴的夹角
l = cos(o);   m = cos(p);    n = sqrt(1-l^2-m^2);      %%方向余弦
```

```
[j,k] = eig(I)      %%本征值与本征矢量
disp(' 转动惯量:')
IL = [l,m,n] * I * [l,m,n]'      %%求转动惯量

n=40;      %%画椭球的数据
phi=(-n:2:n)/n*pi;
theta=(-n:2:n)'/n*pi/2;
x=I(1,1)*cos(theta)*cos(phi);
y=I(2,2)*cos(theta)*sin(phi);
z=I(3,3)*sin(theta)*ones(1,n+1);

for i=2:41      %%将椭球的一些范围的值设为 0 形成剖面图
   if theta(i)>0&theta(i)<pi/2
      y(i,22:31)=0;     x(i,22:31)=0;
   end
end

for i=1:3      %%作坐标变换
   alfa=j(1,i)/j(3,i);
   blta=j(2,i)/j(3,i);
   apha(i,1)=alfa/(sqrt(1+alfa^2+blta^2));
   apha(i,2)=blta/(sqrt(1+alfa^2+blta^2));
   apha(i,3)=1/(sqrt(1+alfa^2+blta^2));
end

xx=[ ];  yy=[ ];   zz=[ ];
for f=1:n+1
   for g=1:n+1
      ass=apha*[x(f,g);y(f,g);z(f,g)];
      xx(f,g)=ass(1);   yy(f,g)=ass(2);   zz(f,g)=ass(3);
   end
end
surf(xx,yy,zz);     %%画椭球
```

2.23 欧拉角

1. 实验题目

演示如何用欧拉角确定定点运动刚体的位置。

2. 实验目的和要求

(1) 演示固连在刚体上的坐标系按欧拉角转动的图像。

(2) 学习指令 rotate 的用法。

(3) 学习对图形句柄操作的指令如 get， gca 等。

3. 题目分析

欧拉角是描述定点运动刚体位置的三个变量。以固定点为原点建立静止坐标系 $O\xi\eta\zeta$，再以固定点为原点建立与刚体固连的运动坐标系 $Oxyz$，确定刚体的位置等价于确定运动坐标系的位置。 Oz 轴的位置用两个夹角确定，一个是 Oz 轴与 $O\zeta$ 轴的夹角 θ，另一个是 Oxy 面和 $O\xi\eta$ 面的交线（称为节线）与 $O\xi$ 轴的夹角 φ，当 Oz 轴的位置确定后，只要 Ox 轴与节线 ON 的夹角定下来，则运动坐标系 $Oxyz$ 的位置就定下来，刚体的位置就定下来了， θ, φ, ψ 称为欧拉角（见图 2.49 ）。三个角的变化范围是 $0 \le \theta \le \pi$; $0 \le \varphi \le 2\pi$; $0 \le \psi \le 2\pi$。

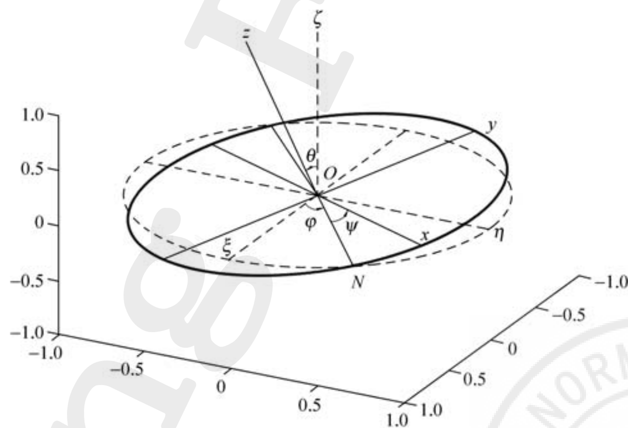


图 2.49 用欧拉角表示的刚体定点转动

程序中演示的顺序是首先使 $O\xi$ 轴绕 $O\zeta$ 轴转动 φ 角，到达节线 ON 的位置；接着 $O\zeta$ 轴绕节线 ON 转动 θ 角到达 Oz 轴的确定位置；最后 Ox 轴绕 Oz 轴转动 ψ 角到达 Ox 轴的最后确定位置。程序中的三个欧拉角的大小可以任意改变，同时可以在窗口任意转动图像，所标注的字母也会随之转动。

4. 思考题

三个欧拉角的转动是否有先后顺序，为什么？

5. 参考程序

主程序的文件名是 elj.m 。

```
a=40;  b=15;  c=60;      %%三个欧拉角
N=100;
t=0:pi/30:2*pi;    L=length(t);
x1=cos(t);    y1=sin(t);    z1=zeros(1,L);     %%画平面上的圆

figure
axis([-1 1 -1 1 -1 1])
hold on
view(116,33)
h1=plot3(x1,y1,z1,'r',[-1,1],[0,0],[0,0],'m',...
    [0,0],[-1,1],[0,0],'r',[0,0],[0,0],[0,1.5],'r');
    %%画开始的 xyz 轴
text(1.1,0,0,'x','fontsize',14)
text(0,1.1,0,'y','fontsize',14)
text(0,0,1.6,'z','fontsize',14)
h1=get(gca,'children');      %%获得当前轴下的全部子对象的句柄

for k=1:N
   A=a/N;
   rotate(h1,[0 90],A,[0 0 0])       %%转 φ 角
   pause(0.1)
end

h2=get(gca,'children');        %%获得当前图形轴下的全部子对象的句柄
xx = get(h2(6),'xdata');       %%获得节线数据
yy = get(h2(6),'ydata');
zz = get(h2(6),'zdata');
plot3(xx,yy,zz,'g')      %%画节线
plot3(x1,y1,z1,'b :',[-1,1],[0,0],[0,0],'k :',...     %%重画原坐标系
   [0,0],[-1,1],[0,0],'b :',[0,0],[0,0],[0,1.5],'b :');
text(1.1,0,0,'\xi','fontsize',14)
```

```
text(0,1.1,0,'\eta','fontsize',14)
text(0,0,1.6,'\zeta','fontsize',14)
aa=a*pi/180;     %%画 φ 角
plot3([0.2,0.2*cos(aa)],[0,0.2*sin(aa)],[0,0])
text(0.25,0.08,0,'\phi','fontsize',14)
pause(1)

for k=1:N
    B=b/N;
    rotate(h2,[a 0],B,[0 0 0])     %%转 θ 角
    pause(0.1)
end

h3=get(gca,'children');
cc=b/180*pi;     %%画 θ 角
plot3([0.2*sin(cc)*sin(aa),0],...
   [-0.2*sin(cc)*cos(aa),0],[0.2*cos(cc),0.3])
text(0.05,-0.06,0.4,'\theta','fontsize',14)
pause(1)

for k=1:N
    C=c/N;
    rotate(h2,[(270+a) (90-b)],C,[0 0 0])     %%转 ψ 角
    pause(0.1)
end
h4=get(gca,'children');
bb=c /180*pi;     %%画 ψ 角
plot3([0.3*cos(aa),0.3*(cos(bb)*cos(aa)-sin(bb)*cos(cc)*sin(aa))],...
    [0.3*sin(aa),0.3*(cos(bb)*sin(aa)+sin(bb)*cos(cc)*cos(aa))],...
    [0,0.3*sin(bb)*sin(cc)])
text(0.25,0.28,0,'\psi','fontsize',14)
```

2.24 圆锥陀螺运动

1. 实验题目

研究拉格朗日——泊松情况下的陀螺运动, 即匀质圆锥陀螺在重力场中的运动。如图 2.50 所示, 设陀螺的顶点为固定点 O , 高为 h , 圆锥底面半径为 R, 密度为 ρ , 则其质量为 $m=\dfrac{\pi}{3}\rho hR^2$, 质心在动力对称轴上, 距固定点 O 的距离为 $l=\dfrac{3}{4}h$ 。

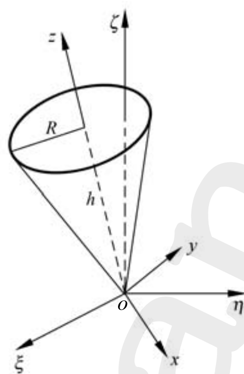


图 2.50 圆锥陀螺的运动

2. 实验目的和要求

(1) 动画演示圆锥陀螺在重力场中的章动和进动。

(2) 画出在不同初始条件下对称轴端点的三种不同的轨迹。

(3) 复习子函数的应用, 本节是在主程序中用子函数来表示坐标变换函数。

(4) 学习用指令 cylinder 画圆锥。

3. 解题分析

以陀螺的固定点 O 为原点建立静止坐标系 $O\xi\eta\zeta$, 以 O 点为原点, 选取陀螺的对称轴为 Oz 轴, 取节线为 Ox 轴, 建立与刚体半固连的主轴坐标系 $Oxyz$, 该坐标系只随陀螺进动和章动, 不随陀螺自转。

陀螺的拉格朗日函数为 $L=T-V$, 动能和势能分别为

$$T=\frac{1}{2}\left(I_x\omega_x^2+I_y\omega_y^2+I_z\omega_z^2\right)$$

$$V=mgl\cos\theta$$

角速度的投影为

$$\left.\begin{aligned}\omega_x &= \frac{\mathrm{d}\theta}{\mathrm{d}t}\\ \omega_y &= \frac{\mathrm{d}\varphi}{\mathrm{d}t}\sin\theta\\ \omega_z &= \frac{\mathrm{d}\varphi}{\mathrm{d}t}\cos\theta+\frac{\mathrm{d}\psi}{\mathrm{d}t}\end{aligned}\right\}$$

式中 θ,φ,ψ 分别为陀螺的章动角、进动角和自转角。由此得

$$L=\frac{1}{2}\left[I_x\left(\frac{\mathrm{d}\theta}{\mathrm{d}t}\right)^2+I_y\left(\frac{\mathrm{d}\varphi}{\mathrm{d}t}\sin\theta\right)^2+I_z\left(\frac{\mathrm{d}\varphi}{\mathrm{d}t}\cos\theta+\frac{\mathrm{d}\psi}{\mathrm{d}t}\right)^2\right]-mgl\cos\theta$$

其中

$$I_x=I_y=\rho\int_0^h\mathrm{d}z\int_0^{2\pi}\mathrm{d}\theta\int_0^{\frac{zR}{h}}(z^2+r^2\sin^2\theta)r\mathrm{d}r=\rho\frac{\pi}{5}h^3R^2+\rho\frac{\pi}{20}hR^4$$

$$I_z=\rho\int_0^h\mathrm{d}z\int_0^{2\pi}\mathrm{d}\theta\int_0^{\frac{zR}{h}}r^3\mathrm{d}r=\rho\frac{\pi}{10}hR^4$$

运动微分方程为

$$\left.\begin{aligned}\frac{\mathrm{d}^2\theta}{\mathrm{d}t^2}&=(1-a)(\frac{\mathrm{d}\varphi}{\mathrm{d}t})^2\sin\theta\cos\theta-a\frac{\mathrm{d}\varphi}{\mathrm{d}t}\frac{\mathrm{d}\psi}{\mathrm{d}t}\sin\theta+bg\sin\theta\\ \frac{\mathrm{d}^2\varphi}{\mathrm{d}t^2}&=(a-2)\frac{\mathrm{d}\theta}{\mathrm{d}t}\frac{\mathrm{d}\varphi}{\mathrm{d}t}\ \mathrm{ctg}\theta+\frac{a}{\sin\theta}\frac{\mathrm{d}\theta}{\mathrm{d}t}\frac{\mathrm{d}\psi}{\mathrm{d}t}\\ \frac{\mathrm{d}^2\psi}{\mathrm{d}t^2}&=\frac{\mathrm{d}\theta}{\mathrm{d}t}\frac{\mathrm{d}\varphi}{\mathrm{d}t}[\sin\theta-(a-2)\ \mathrm{ctg}\theta\cos\theta]-a\frac{\mathrm{d}\theta}{\mathrm{d}t}\frac{\mathrm{d}\psi}{\mathrm{d}t}\ \mathrm{ctg}\theta\end{aligned}\right\}\qquad(2.24.1)$$

其中

$$a=\frac{I_z}{I_x}=\frac{2}{4h^2/R^2+1};\ b=\frac{ml}{I_x}=\frac{5\rho h}{4h^2+R^2}$$

令 $y_1=\theta,\ y_2=\mathrm{d}\theta/\mathrm{d}t,\ y_3=\varphi,\ y_4=\mathrm{d}\varphi/\mathrm{d}t,\ y_5=\psi,\ y_6=\mathrm{d}\psi/\mathrm{d}t$ 得

$$\left.\begin{aligned}\frac{\mathrm{d}y_1}{\mathrm{d}t}&=y_2\\ \frac{\mathrm{d}y_2}{\mathrm{d}t}&=(1-a)(y_4)^2\sin y_1\cos y_1-ay_4y_6\sin y_1+bg\sin y_1\\ \frac{\mathrm{d}y_3}{\mathrm{d}t}&=y_4\\ \frac{\mathrm{d}y_4}{\mathrm{d}t}&=(a-2)y_2y_4\ \mathrm{ctg}y_1+\frac{a}{\sin y_1}y_2y_6\\ \frac{\mathrm{d}y_5}{\mathrm{d}t}&=y_6\\ \frac{\mathrm{d}y_6}{\mathrm{d}t}&=y_2y_4[\sin y_1-(a-2)\ \mathrm{ctg}y_1\cos y_1]-ay_2y_6\ \mathrm{ctg}y_1\end{aligned}\right\}$$

这个微分方程的解法很简单，主要的问题是如何作出动画。从 2.23 节欧拉角的介绍中可知，如果以圆锥的顶点为原点建立一个静止坐标系，再以圆锥的顶点为原点建立一个与刚体固连的动坐标系，则圆锥表面上一点的坐标 x_0, y_0, z_0 在动坐标系中是一个常数，而这点在静止坐标系中的坐标 x, y, z 则可以通过欧拉角表示的坐标变换来得到，从几何关系可以得出，这个坐标变换关系是

$$\left.\begin{aligned}
x &= x_0(\cos\psi\cos\varphi - \sin\psi\cos\theta\sin\varphi)+\\
&\quad\; y_0(-\sin\psi\cos\varphi - \cos\psi\cos\theta\sin\varphi) + z_0\sin\theta\sin\varphi\\
y &= x_0(\cos\psi\cos\varphi + \sin\psi\cos\theta\cos\varphi)+\\
&\quad\; y_0(-\sin\psi\sin\varphi + \cos\psi\cos\theta\cos\varphi) + z_0(-\sin\theta\cos\varphi)\\
z &= x_0\sin\psi\sin\theta + y_0\cos\psi\sin\theta + z_0\cos\theta
\end{aligned}\right\} \tag{2.24.2}$$

若将从微分方程中解得的欧拉角随时间变化的规律 $\theta = \theta(t), \varphi = \varphi(t), \psi = \psi(t)$ 代入，就得出此点相对静止坐标系的运动规律。在主程序中，按照这个公式建立了一个子函数来计算运动中圆锥表面上各点在静止坐标系中的坐标。

4. 思考题

请自己推导微分方程组 (2.24.1) 和变换关系 (2.24.2) 。

5. 参考程序

主程序的文件名是 tlyd.m 。

```
function tlyd
axis([-1 1 -1 1 0 1]);
hold on;
text(0,2,0.8,' 陀螺运动 ','fontsize',40,'color','r');
grid on;
[x0,y0,z0]=cylinder(0:.05:.5,60);       %%陀螺侧面的数据网格
Q=linspace(0,2*pi,60);      %%以下四句是陀螺上表面的数据网格
cx0=0.5*cos(Q);
cy0=0.5*sin(Q);
cz0=ones(1,60);
phi = 0; thi = pi/6; psi = 0;
[x,y,z] = zbbh(x0,y0,z0,thi,phi,psi);      %%陀螺初始位置侧面数据
[cx,cy,cz] = zbbh(cx0,cy0,cz0,thi,phi,psi);  %%陀螺初位置上表面数据
```

```
h1= surf(x,y,z);      %%画初始位置陀螺侧面
h2=fill3(cx,cy,cz,[0.8 0.8 0.8]);     %%画初始位置陀螺的上表面
pause(0.5);

R = 1; h = 2; p = 1; g=9.8;     %%陀螺的参数
a=2/(4*h^2/R^2+1);  b=5*p*h/(4*h^2+R^2);
[t,u] = ode45('tlydfun',[0:0.1:25],[thi 0 phi 0 psi 75],[ ],a,b,g);
for i = 1 :  length(u);
         %%求陀螺新位置侧面数据
   [x,y,z] = zbbh(x0,y0,z0,u(i,1),u(i,3),u(i,5));
        %%求陀螺新位置上表面数据
   [cx,cy,cz] = zbbh(cx0,cy0,cz0,u(i,1),u(i,3),u(i,5));
    set(h1,'xdata',x,'ydata',y,'zdata',z);     %%画新位置陀螺
    set(h2,'xdata',cx,'ydata',cy,'zdata',cz);
    drawnow;
end;
text(-1.4,0,' 结束 ','fontsize',40,'color','r');
function [x,y,z] = zbbh(x0,y0,z0,thi,phi,psi)     %%坐标变换子函数
x=x0*(cos(psi)*cos(phi)-sin(psi)*cos(thi)*sin(phi))...
   +y0*(-sin(psi)*cos(phi)-cos(psi)*cos(thi)*sin(phi))...
   +z0*sin(thi)*sin(phi);
y=x0*(cos(psi)*sin(phi)+sin(psi)*cos(thi)*cos(phi))...
   +y0*(-sin(psi)*sin(phi)+cos(psi)*cos(thi)*cos(phi))...
   -z0*sin(thi)*cos(phi);
z=x0*sin(psi)* sin(thi)+y0*cos(psi)*sin(thi)+z0*cos(thi);
```

函数文件是一个独立的文件，文件名为 tlydfun.m 。

```
function y = tlydfun(t,u,flag,a,b,g)
y = [u(2);
    (1-a)*u(4)^2*sin(u(1))*cos(u(1))...
        -a*u(4)*u(6)*sin(u(1))+b*g*sin(u(1));
     u(4);
    (a-2)*u(2)*u(4)*cot(u(1))+a*u(2)*u(6)/sin(u(1));
     u(6);
     u(2)*u(4)*(sin(u(1))-(a-2)*cot(u(1))*cos(u(1)))...
        -a*u(2)*u(6)*cot(u(1))];
```

附录A　MATLAB 主要指令

A0　主要指令分类

general	通用指令	graphics	通用绘图函数
ops	运算符和特殊算符	uitools	图形用户界面工具
lang	程序语言结构和调试指令	strfun	字符串函数
elmat	基本矩阵和矩阵操作	iofun	底层文件输入 / 输出函数
elfun	基本数学函数	timefun	时间和日期
specfun	特殊数学函数	datatypes	数据类型和结构
matfun	矩阵函数和数值线性代数	winfun	窗口操作系统界面文件
datafun	数据分析和傅里叶变换	sounds	声音处理函数
polyfun	多项式与插补函数	dde	动态数据交换函数
funfun	非线性数值功能函数	local	主启动文件
sparfun	稀疏矩阵函数	demos	演示函数
graph2d	二维图形	pde	偏微分方程工具箱
graph3d	三维图形	symbolic	符号运算工具箱
specgraph	特殊作图函数		

A1　常用指令(General Purpose Commands)

A1.1　管理指令和函数(Managing Commands and Functions)

addpath	添加搜索路径	path	控制搜索路径
doc	显示超文本帮助	pathtool	路径管理器
help	在线帮助指令	profile	运行记录
helpdesk	在线帮助桌面	profreport	产生运行记录的报告
helpwin	在线帮助窗口	rmpath	删除某搜索目录
lasterr	最后的错误信息	type	显示文件内容
lastwarn	最后的警告信息	web	给出浏览器或主页地址
lookfor	关键词检索	whatsnew	显示 README 文件
partialpath	局部路径名	which	确定指令和文件的位置
what	列出 m, mat, mex 文件		

A1.2　变量和内存空间的管理(Managing Variab1es and the Workspace)

clear	从内存中清除变量和函数	pack	合并工作内存中的碎块
disp	显示矩阵和文字内容	save	把内存变量存入磁盘
1ength	确定矢量的长度	saveas	存储图形或模型
1oad	从磁盘中调入数据变量	size	确定矩阵的维数
mlock	使 M 文件免删除	who,whos	列出工作内存中的变量名
munlock	允许 M 文件被删除	workspace	工作内存浏览器

openvar	打开工作内存中的变量

A1.3 指令窗控制(Controlling the Command Window)

clc	清除指令窗口	echo	执行 M 文件时显示的切换开关
home	光标返回行首	more	指令窗分页输出的控制开关
format	设置数据输出格式		

A1.4 调用操作系统和文件处理(Working with Files and the Operating System)

cd	改变当前工作目录	inmem	内存中的函数
copyfile	复制文件	matlabroot	MATLAB 所在的根目录
delete	删除文件	mkdir	建立目录
diary	储存工作窗操作内容	open	打开文件
dir	列出的文件	pwd	显示当前目录
edit	矩阵编辑器	tempname	临时文件名
fileparts	文件的路径、名称等	!	执行外部应用程序
fullfile	文件的全部信息		

A1.5 MATLAB 的启动和终止(Starting and Quitting from MATLAB)

matlabrc	MATLAB 的主启动文件	startup	启动时自动执行 M 文件
quit	退出 MATLAB		

A2 运算符和特殊算符(Operators and Special Characters)

A2.1 常用符号(Operators and Special Characters)

+	加	.	小数点
-	减	···	续行号
*	矩阵乘	,	逗号
.*	数组乘	;	分号
^	矩阵乘方	%	百分号
.^	数组乘方	!	惊叹号
\	左除	′	转置号和引号
/	右除	.′	非共轭转置
./ .\	数组右（左）除	=	赋值符号
kron	张量积	==	等号
:	冒号	&	逻辑与
( )	圆括号	|	逻辑或
[ ]	方括号	~	逻辑非
{ }	花括号	xor	逻辑异或

A2.2 逻辑操作(Logical Characteristics)

all	全非零元矢量为真	is *	查询状态
any	有非零元的矢量为真	isa	查询对象的分类
exist	检查变量或函数是否被定义	logical	把数值型转为逻辑型
find	找出非零元素的下标	mislocked	当 M 文件不能被删除时为真

A3 语言结构和调试指令(Language Constructs and Debugging)

A3.1　编程语言(MATLAB as a Programming Language)

builtin	用超载方法运行内部函数	function	函数文件头
eval	字符串宏指令	global	定义全局变量
evalc	对输入的表达式求值	nargchk	输入变量个数检查
evalin	对工作内存中的表达式求值	persistent	定义永久变量
feval	函数宏指令	script	命令文件

A3.2　控制语句(Control Flow)

break	终止最内循环	if	条件执行语句
case	case 选择	return	返回
else	同 IF 一起使用	switch	开关
elseif	同 IF 一起使用	try	开始 try 块
end	结束 FOR 、 WHILE 、 IF 语句	warning	显示警告信息
error	显示错误信息	while	不确定次数重复执行语句
for	按规定次数重复执行语句		

A3.3　交互式输入(Interactive input)

input	提示键盘输入	menu	创建菜单
keyboard	激活键盘作为命令文件	pause	暂停

A3.4　面向对象编程(Object-oriented programming)

class	创建新类或返回对象的类	int8(16,32)	转为符号整数
double	转换为双精度型	saveobj	存储对象的滤波器
inferiorto	类关系运算符	single	转为单精度型
inline	创建在线对象	superiorto	类关系运算符
loadobj	为用户对象扩充的下载函数	unit8(16,32)	转为 8,16,32 位型

A3.5　调试指令(Debugging Commands)

dbclear	清除断点	dbstack	列出调用堆栈
dbcont	恢复执行	dbstatus	列出所有的断点
dbdown	改变局部工作内存的前后关系	dbstep	单步执行
dbmex	调试 MEX 文件	dbstop	设置断点
dbquit	结束调试模式	dbtype	列出带行号的 M 文件
dbup	改变局部工作内存的前后关系		

A4　基本矩阵函数和操作(Elementary Matrices and Matrix Manipulation)

A4.1　基本矩阵(Elementary Matrices)

blkdiag	用输入量建立块对角矩阵	rand	均匀分布随机阵
eye	单位阵	randn	正态分布随机阵
linspace	线性等分矢量	zeros	全零矩阵
logspace	对数等分矢量	:	生成等间距矢量
ones	全 1 矩阵		

A4.2　特殊变量和常数(Special Variables and Constants)

ans	最新表达式的运算结果	nargin	函数输入参量的个数

computer 计算机类型
eps 浮点相对误差
flops 浮点运算次数
i, j 虚数单位
Inf 无穷大
inputname 输入变量名
NaN 非数
nargout 函数输出参量的个数
pi 3.1415926535897 . . .
realmax 最大浮点数
realmin 最小正浮点数
varargin 输入变量数
varargout 输出变量数

A4.3 时间和日期(Time and Dates)

cakendar 日历
clock 时钟
cputime MATLAB 占用 CPU 时间
date 日期
datenum 日期数字系列
datestr 日期字符串格式
datevec 日期分量
eomday 月末
etime 计算机内部时钟
tic 秒表起动
toc 秒表终止和显示
weekday 星期几

A4.4 矩阵操作(Matrix Manipulation)

cat 数组组合
diag 创建对角阵，抽取对角矢量
fliplr 矩阵的左右翻转
flipud 矩阵的上下翻转
repmat 复制和编排矩阵
reshape 矩阵变维
rot90 矩阵逆时针旋转 90°
triI 抽取下三角阵
triu 抽取上三角阵
: 矩阵的援引和重排

A4.5 特殊矩阵(Specialized Matrices)

compan 伴随矩阵
gallery 一些小测试矩阵
hadamard Hadamard 矩阵
hankel Hankel 矩阵
hilb Hilbert 矩阵
invhilb 逆 Hilbert 矩阵
magic 幻方阵
pascal Pascal 矩阵
toeplitz Toeplitz 矩阵
wilkinson Wilkinson's 特征值矩阵

A5 基本数学函数(Elementary Math Functions)

A5.1 三角和超越函数(Trigonometric and Hyperbolic)

abs 绝对值
acos,acosh 反余弦，反双曲余弦
acot ,acoth 反余切，反双曲余切
acsc,acsch 反余割，反双曲余割
asec, asech 反正割，反双曲正割
asin, asinh 反正弦，反双曲正弦
atan, atanh 反正切，反双曲正切
atan2 四象限反正切
ceil 向正无穷大取整
complex 由实部和虚部构造虚数
conj 取复共轭
floor 朝负无穷取整
gcd 最大公因子
imag 复数虚部
lcm 最小公倍数
log 自然对数
log2 以 2 为底的对数
log10 常用对数
mod 求余
nchoosek 二项式系数
real 复数的实部
rem 除法的余数

cos, cosh	余弦，双曲余弦	round	四舍五入取整
cot, coth	余切，双曲余切	sec, sech	正割，双曲正割
csc,csch	余割，双曲余割	sin, sinh	正弦，双曲正弦
exp	指数	sqrt	平方根
fix	朝零取整	tan, tanh	正切，双曲正切
sign	符号函数		

A5.2　特殊数学函数 (Specialized Math Functions)

airy	Airy 函数	erf,erfc,erfcx	误差函数值
besselh	Hankel 函数	expint	指数积分函数
besseli, besselk	改进的 Bessel 函数	factorial	阶乘函数
besselj, bessely	Bessel 函数	gamma,gammainc	Gamma 函数
beta, betainc	Beta 函数	gammaln	Gamma 函数对数
betaln	Beta 函数的对数	legendre	Legendre 函数
ellipkj	Jacobi 椭圆函数	pow2	求 2 的幂
ellipke	完全的椭圆函数	rat rats	有理近似

A6　坐标系统转换(Coordinate System Conversion)

cart2pol	直角坐标到极坐标的转换	pol2cart	极坐标到直角坐标的转换
cart2sph	直角坐标到球坐标的转换	sph2cart	球坐标到直角坐标的转换

A7　矩阵函数和数值线性代数 (Matrix Functions — Numerical Linear Algebra)

A7.1　矩阵分析 (Matrix Analysis)

cond	矩阵条件数	rank	秩
condeig	和本征值相关的条件数	rcond	逆条件数
det	行列式的值	rref,rrefmovie	转换为行阶梯形
norm	矩阵或矢量范数	subspace	两个子空间的角度
null	零空间	trace	迹
orth	正交化空间		

A7.2　线性方程(Linear Equations)

chol	Cholesky 分解	lsqnonneg	非负最小二乘解
inv	矩阵的逆	pinv	伪逆
lscov	已知协方差的最小二乘解	qr	QR 分解
lu	LU 分解		

A7.3　本征值与奇异值 (Eigenvalues and Singular Values)

balance	改善特征值精度的平衡刻度	qz	广义特征值
cdf2rdf	复数对角型转换到实块对角型	rsf2csf	实对角型转成复对角型
eig	矩阵特征值和特征矢量	schur	Schur 分解
hess	Hessenberg 矩阵	svd	奇异值分解
poly	求指定根的特征多项式		

A7.4　矩阵函数 (Matrix Functions)

expm	矩阵指数	logm	矩阵对数

funm	计算一般矩阵函数	sqrtm	矩阵平方根

A7.5 低级函数

qrdelete	从 QR 分解中删除列	qrinsert	从 QR 分解中加入列

A8 数据分析和傅里叶变换(Date Analysis and Fourier Transform Functions)

A8.1 基本运算(Basic Operations)

convhull	凸壳函数	perms	所有可能的排序
cumprod	累计积	polyarea	多边形的面积
cumsum	累计和	primes	生成质数列表
cumtrapz	累计梯形积分	prod	元素积
delaunay	Delaunay 三角化	sort	由小到大排序
dsearch	求最近点	sortrows	将行按升序排列
factor	质数分解	std	标准差
inpolygon	搜索多边形内的点	sum	元素和
max	最大值	trapz	梯形数值积分
mean	平均值	tsearch	搜索 Delaunay 三角形
median	中值	var	偏差值
min	最小值	voronoi	Voronoi 图

A8.2 有限差分(Finite differences)

de12	离散 Laplacian 算子	gradient	梯度
diff	差分和近似微分		

A8.3 相关(Correlation)

corrcoef	相关系数	cov	协方差矩阵

A8.4 滤波和卷积 (Filtering and Convolution)

conv	卷积和多项式相乘	filter	一维数字滤波器
conv2	二维卷积	filter2	二维数字滤波器
deconv	解卷和多项式相除		

A8.5 傅里叶变换(Fourier Transform)

abs	幅值	ifft	离散傅里叶反变换
angle	相角	ifft2	二维离散傅里叶反变换
cplxpair	复数阵成共轭对形式排列	ifftn	多维离散傅里叶反变换
fft	快速离散傅里叶变换	ifftshift	逆 fftshift 移动
fft2	二维离散傅里叶变换	nextpow2	最近邻的 2 的幂
fftn	多维离散傅里叶反变换	unwrap	相位角 360° 线调整
fftshift	将 fft 的直流分量移到谱中心		

A8.6 矢量运算(Vector Operations)

cross	矢量叉积	setxor	求两个矢量的异或
intersect	两个矢量的交	union	求两个矢量的并
ismember	检查集合的元素	unique	返回矢量的元素值
setdiff	两个集合的差集		

A9　多项式与插补函数(Polynomial and Interpolation Functions)

A9.1　多项式(Polynomials)

conv	多项式相乘	plyval	求多项式的值
deconv	多项式相除	polyvalm	求矩阵多项式的值
poly	由根创建多项式	residue	求部分分式表达
polyder	多项式微分	roots	求多项式的根
polyfit	多项式拟合		

A9.2　数据插补(Data Interpolation)

griddata	三维分格点数据	interpn	多维插补
interp1	一维插补	meshgrid	产生三维图形的 XY 矩阵
interp2	二维插补	ndgrid	产生多维函数和插补的数据
interp3	三维插补	spline	三次样条插补
interpft	利用 FFT 方法一维插补		

A10　非线性数值功能函数(Function Functions—NonLinear Nnmerical Methods)

dblquad	二重积分	odefile	定义微分方程问题
fminbnd	单变量函数最小值	odeget	获取选项设置的内容
fminsearch	多变量函数最小值	odeset	解微分方程的选项
fzero	单变量函数的零点	quad, quad8	数值积分
ode23,ode45,ode113	解非刚性微分方程	vectorize	将表达式矢量化
ode15s,ode23s	解刚性微分方程	ode23t,ode23tb	解刚性微分方程

A11　稀疏矩阵函数(Sparse Matrix Functions)

A11.1　基本稀疏矩阵(Element Sparse Matrices)

spdiags	从对角阵形成稀疏阵	sprandn	正态分布稀疏随机阵
speye	稀疏单位阵	sprandsym	稀疏对称随机阵
sprand	稀疏随机阵		

A11.2　全阵与稀疏阵的转换(Full to Sparse Conversion)

find	寻找非零元素下标	sparse	创建稀疏阵
full	把稀疏阵转换成全元素阵	spconvert	把稀疏阵转换到指定格式

A11.3　稀疏矩阵非零元的处理(Working with Nonzero Matrices of Sparse Matrices)

nnz	非零元素个数	spalloc	为非零元素分配存储器
nonzeros	非零元素	spfun	非零元素进行函数计算
nzmax	为非零元素分配的存储器数	spones	用 1 代替非零元素

A11.4　稀疏矩阵的可视化(Visualizing Sparse Matrices)

Spy　稀疏矩阵的图形表示

A11.5　排序算法 (Reordering Algorithms)

colmmd	列最小度排序	randperm	随机置换矢量
colperm	基于非零算法列排序	symmmd	对称最小度排序
dmperm	Dulmage-Mendelsohn 分解	symrcm	反向 Cuthi11-McKee 排序

A11.6　范数、条件数和秩(Norm,condition Number, and Rank)

condest	估计范 1 条件数	normest	估计 2 范数

A11.7 线性方程的稀疏系统 (Sparse Systems of Linear Equations)

bicg	双共轭梯度法	pcg	预处理共轭梯度法
bicgstab	双共轭梯度稳定法	qmr	Quasi-Minimal 残差法
cgs	二次共轭梯度法	qr	正交三角分解
cholinc	不完全 Cholesky 分解	qrdelete	删除 QR 分解中的列
cholupdate	一阶 Cholesky 分解更新	qrinsert	插入 QR 分解中的列
gmres	标准化最小残差法	qrupdate	一阶 QR 分解更新
luinc	不完全 LU 分解		

A11.8 稀疏矩阵本征值和奇异值 (Sparse Eigenvalues and Singular Values)

eigs	求本征值和本征矢量	svds	求奇异值

A11.9 关于稀疏矩阵的其他指令(Miscellaneous)

spparms	对稀疏矩阵程序设置参数

A12 声音处理函数 (Sonnd Processing Functions)

A12.1 常用声音函数 (Generation Sound Functions)

lin2mu	把线性声音信号转变为 mu-law 信号	sound	把矢量转换成声音
mu2lin	把 mu-law 信号转变为线性	soundsc	标度数据并作为声音演播

A12.2 特殊声音函数 (Sparcstation-Specific Sound Functions)

auread	读 AU 文件	auwrite	写 AU 文件

A12.3 处理 WAV 声文件(WAV Sound Functions)

wavread	读 wave 文件	wavwrite	写 wave 文件

A13 字符串函数(Character String Functions)

A13.1 通用字符串函数 (General)

abs	绝对值	real	复数实部
eval	执行串形式的 MATLAB 表达式	strings	字符串句柄

A13.2 字符串操作(String Manipulation)

deblank	删除串最后的空格	strmatch	寻找字符串可能的匹配
findstr	在一个串中寻找一个子串	strncmp	比较字串的前 N 个字母
lower	把字符串变成小写	strrep	替代一个串中的子串
strcat	字符串组合	strtok	删除串中的指定子串
strcmp	比较字符串	strvcat	垂直合并字符串
strcmpi	不管大小写比较字符串	symvar	在表达式中确定符号变量
strjust	判断字符矩阵	upper	把字符串变成大写

A13.3 字符串与数值间的转换(String to Number Conversion)

char	建立字符矩阵	sprintf	按格式把数字转换为串
int2str	把整数转换为串	sscanf	按格式把串转换为数字
mat2str	把矩阵转换为字符串	str2double	把字符串转为双精度数
num2str	把浮点数转换为串	str2num	把串转换为浮点数

A13.4 数制间的转换 (Radix Conversion)

bin2dec	把 2 进制整数变成 10 进制	hex2dec	把 16 进制串变成 10 进制数
dec2bin	把 10 进制整数变成 2 进制	hex2num	把 16 进制串变成双精度数
dec2hex	把 10 进制整数变成 16 进制		

A14　低层文件输入输出函数 (Low-level File I/O Functions)

A14.1　文件的开启和关闭(File opening and Closing)

fclose	关闭文件	fopen	打开文件

A14.2　无格式输入输出(Unformatted I/O)

fread	从文件中读二进制数据	fwrite	把二进制数据写入文件

A14.3　有格式输入输出 (Formatted I/O)

fget1	按行从文件读取数据并清除文件指针	fprintf	把格式化数据写入文件
fgets	按行从文件读取数据并保留文件指针	fscanf	从文件中读格式化数据

A14.4　文件定位 (File Positioning)

feof	测试文件是否结束	fseek	设置文件指针
ferror	查询文件输入／输出错误状态	ftell	得到文件指针
frewind	反绕文件		

A14.5　串的转换(String Conversion)

sprintf	写格式数据到串	sscanf	在格式控制下读串

A14.6　特殊 I/O 文件 (Specialized File I/O)

dlmread	把 ASCII 中的数据读入矩阵	imwrite	将图形写入图形文件
dlmwrite	把矩阵写入 ASCII 文件	textread	从文本文件读格式化数据
hdf	HDF 界面	wklread	将 LOTUS-123 文件读入矩阵
imfinfo	返回图形文件信息	wklwrite	将矩阵写入 LOTUS-123 文件
imread	从图形文件读出图形数据		

A15　位函数(Bitwise Fuctions)

bitand	位和	bitset	给位赋值
bitcmp	补码	bitshift	移位
bitor	位或	bitget	获取位
bitmax	机器的最大浮点整数	bitxor	位异或

A16　结构数组函数 (Structure Functions)

deal	处理输入输出	setfield	设置结构的属性值
fieldnames	结构的属性名称	struct	生成结构数组
getfield	获取结构的属性	struct2cell	把结构数组转为基元数组
rmfield	删除结构的属性		

A17　对象函数(Object Functions)

class	生成对象或返回对象的类名	isa	检验对象是否属于某类

A18　基元数组函数(Cell Array Functions)

cell	生成基元数组	celldisp	显示基元数组
cellfun	对基元的每个元素应用函数	cellplot	用图形显示基元数组
cellstr	用字符数组生成基元数组	num2cell	把数值数组转换为基元数组

cell2struct 将基元数组转换为结构数组

A19 多维数组函数 (Multidimensional Array Functions)

cat	数组组合	permute	数组重组
flipdim	按指定的维翻转数组	reshape	数组变形
ind2sub	数组的下标转换	shiftdim	维数移位
ipermute	数组维数的逆变换	squeeze	删除数值为 1 的维数
ndgrid	为多维函数和插值生成数据	sub2ind	单个下标
ndims	数组的维数		

A20 作图和数据可视化(Plotting and Data Visualization)

A20.1 基本图形 (Basic Plots and Graphs)

bar	垂直直方图	plot	直角坐标下画曲线
barh	水平直方图	polar	极坐标曲线图
hist	统计频数直方图	semilogx	X 轴半对数刻度曲线
hold	保持目前的图形	semilogy	Y 轴半对数刻度曲线
loglog	双对数刻度曲线	subplot	分区作图
pie	扇形图		

A20.2 三维图形函数(Three-Dimensional Plotting)

bar3	垂直三维直方图	quiver	二维矢量场图
bar3h	水平三维直方图	quiver3	三维矢量场图
comet3	三维彗星动态轨迹线图	slice	切片图
cylinder	生成圆柱面	sphere	生成球面
fill3	三维曲面多边形填色	stem3	画分离的表面数据
plot3	三维直角坐标曲线图	waterfall	瀑布水线图

A20.3 图形注解和网格 (Plot Annotation and Grids)

clabel	给等高线加标注	plotyy	在左（右）边画 Y 轴刻度
datetick	生成刻度线数据	title	图形标题
grid	画坐标网格线	xlabel	X 轴名标注
gtext	用鼠标在图上标注文字	ylabel	Y 轴名标注
legend	图例说明	zlabel	Z 轴名标注
plotedit	图形编辑		

A20.4 表面图、网线图和等高线图 (Surface, Mesh and Contour Plots)

contour	等高线图	surf	三维表面图
contourc	等高线计算	surface	建立低层对象的表面
contourf	等高线填色	surfc	带等高线的三维表面图
hidden	隐藏网格线	surfl	带光照的三维表面图
meshc	带等高线的三维网线图	trimesh	三角网格图
mesh	三维网线图	trisurf	三角表面图
peaks	一个两变量函数		

A20.5 立体图(Volume Visualization)

coneplot	画三维速度场中的速度矢量	shrinkface	减小块表面的大小
contourslice	在立体切片图中画等高线	smooth3	平滑 3 维数据
isocaps	计算等位面	stream2	计算 2 维流线数据
isonormals	计算等位面顶点法线	stream3	计算 3 维流线数据
isosurface	从体数据中提取等位面数据	streamline	画流线图
reducepatch	减小块表面的数目	surf2patch	表面数据转为块数据
reducevolume	在体数据中减小元素数	subvolume	从体数据中提取数据

A20.6　产生区域数据 (Domain Generation)

griddata	数据栅格化和表面拟合	meshgrid	为三维图产生 XY 数据

A20.7　特殊作图(Specialized Plotting)

area	面积图	feather	复数矢量图
box	二维或三维图的坐标轴框	fplot	函数曲线图
comet	二维彗星状轨迹图	inpolygon	点在多边形内为真
compass	从原点出发的复数矢量图	pareto	PARETO 图
convhull	凸壳函数	pcolor	用颜色反映数据的伪色图
delaunay	DELAUNAY 三角形化	pie3	三维饼图
dsearch	搜索最近的 DELAUNAY 三角形	plotmatrix	用矩阵作图
errorbar	误差棒图	polyarea	多边形面积
ezcontur	快速画等高线	quiver	矢量场图
ezcintourf	快速等高线填色	ribbon	RIBBON 图
ezmesh	快速作三维网格	rose	统计频数扇形图
zemeshc	快速三维网格和等高线的结合	scatter	二维的圆点图
ezplot	快速函数作图	scatter3	三维的圆点图
ezplot3	快速三维参数作图	stairs	阶梯形曲线图
ezpolar	快速极坐标图	stem	火柴杆图
ezsurf	快速三维表面图	tsearch	搜索 Delaunay 三角形
ezsurfc	快速三维表面和等高线图	voronoi	VORONOI 图

A20.8　视角控制(View Control)

camdolly	移动相机和目标的位置	camva	设置或获取相机的视角
camlookat	观察特定的目标	camzoom	放大或缩小相机的视野
camorbit	相机目标的轨迹	daspect	设置或获取数据各向比例
campan	绕相机位置旋转目标	phaspect	设置或获取图形的各向比例
campos	设置或获取相机位置	view	三维图的视点
camproj	设置或获取投影方式	viewmtx	产生视角变换矩阵
camroll	绕观察轴旋转相机	xlim	设置或获取 X 轴的范围
camtarget	设置或获取相机目标	ylim	设置或获取 Y 轴的范围
camup	设置或获取相机的顶矢量	zlim	设置或获取 Z 轴的范围

A20.9　光照模式 (Lighting)

camlight	产生或定位光照	lighting	光照模式

lightangle 光照的球状定位 material 物质反射模式

A20.10 颜色操作 (Color Operations)

brighten	控制色彩的明暗	hsv2rgb	饱和色彩数据转换红绿蓝数据
caxis	彩色轴标度	rgb2hsv	红绿蓝数据转换饱和色彩数据
colorbar	显示颜色对应的数值	shading	图形渲染模式
colordef	设置颜色默认值	spinmap	颜色周期性变化操纵
colormap	设置色图	surgnorm	三维表面法线
graymon	单色灰度图的默认商设置	whitebg	设置图形窗口为白底
rgbplot	取色曲线图		

A20.11 彩色图 (colormap)

autumn	红黄色调图	hsv	饱和色彩图
bone	蓝色调灰度图	jet	变异 HSV 色图
contrast	提高图像对比度的灰色图	lines	线性色度图
cool	青和品红浓淡色图	prism	光谱色图
copper	线性变化纯铜色调图	spring	洋红和黄色调图
flag	红白蓝黑交错色图	summer	绿黄色调图
gray	线性灰度色图	winter	蓝绿色调图
hot	黑红黄白交错色图	pink	粉红色图
colorcube	增强的立方色图		

A20.12 打印(Printing)

orient	设置走纸方向	printopt	打印机设置
print	打印图形或把图存入文件	saveas	存储图形文件

A20.13 一般的图形句柄 (Handle Graphics, General)

copyobj	复制对象	get	获得对象属性
findobj	用规定的特性找寻对象	ishandle	对图形对象为真
gcbo	获得当前对象的柄	rotate	在给定方向上旋转对象
gco	返回当前对象的句柄	set	建立对象特性

A20.14 产生图形对象的句柄(Handle Graphics, Object Creation)

axes	创建轴对象	rectangle	创建二维矩形对象
figure	创建图形窗口	surface	创建面
image	创建图像	text	创建图形中文本
line	创建线	unicontextmenu	创建目录菜单
patch	创建块		

A20.15 产生图形窗口的句柄 (Handle Graphics, Figure Windows)

capture	当前图的屏捕捉	gcf	获得当前图的柄
clc	清除图形窗口	newplot	下一个新图
clf	清除当前图	refresh	更新图像
close	关闭图形	saveas	存储图形

A20.16 产生轴的句柄 (Handle Graphics, Axes)

axis	轴的控制	gca	获得当前轴的柄
cla	清除当前轴		

A20.17　对象操作 (Object Manipulation)

reset	重设对象特性	rotate3d	旋转对象

A20.18　交互输入 (Interactive User Input)

ginput	从鼠标得到图形点坐标	zoom	改变二维图像大小

A20.19　操作区(Region of Interest)

rbbox	擦除框	dragrect	用鼠标拖动 XOR 矩形框
drawnow	屏幕刷新		

A21　产生用户界面(Graphical User Interface Creation)

A21.1　对话框 (Dialog Boxes)

dialog	建立对话框	printdlg	显示打印对话框
errordlg	建立误差对话框	questdlg	建立问题对话框
helpdlg	显示帮助对话框	uigetfile	打开一个文件查询对话框
inputdlg	建立输入对话框	uiputfile	打开一个文件存储对话框
listdlg	建立列表对话框	uisetcolor	用对话框交互设置颜色
msgbox	建立信息对话框	uisetfont	用对话框交互设置字体
pagedlg	建立页面设计对话框	warndlg	建立警告信息对话框

A21.2　用户对象界面 (User Interface Objects)

uicontrol	建立用户界面控制	menu	建立用户输入选择对话框
uicontextmenu	建立目录菜单	uimenu	建立用户使用菜单控制

A21.3　其它指令 (Other Functions)

dragrect	用鼠标拖动矩形框	selectmoveresize	选择、移动、更改对象大小
gcbo	返回对象句柄	textwrap	对给定的控制返回打包的字串
getframe	获得影片动画图像的帧	uiresume,uiwait	同时使用，控制程序的执行
movie	播放影片动画	waitbar	显示等待工具条
moviein	影片动画内存初始化	waitforbuttonpress	等待键盘命令
rbbox	在选定区域建立擦除框		

附录B　符号工具箱指令

B1 微积分计算 (Calculus)

diff	微分	taylor	泰勒展开
int	积分	jacobian	约当矩阵
limit	求极限	symsum	求级数和

B2 线性代数 (Linear Algebra)

diag	建立或提取对角元	null	空矩阵的基
triu	上三角矩阵	colspace	列空间的基
tril	下三角矩阵	eig	求本征值和本征矢量
inv	求逆矩阵	svd	奇异值分解
det	行列式的值	jordan	约当标准型
rank	矩阵的秩号	poly	特征多项式
rref	简化梯形阵	expm	矩阵指数函数

B3 化简 (Simplification)

simplify	函数化简	numden	找表达式的分子分母
expand	函数展开	horner	嵌套多项式表示
factor	分解因式	subexpr	用一个符号表示表达式中共同的部分重写表达式
collect	合并同类项		
simple	简化表达式	subs	符号变量的替代

B4 解方程 (Solution of Equations)

solve	解方程	finverse	求反函数
dsolve	解微分方程	compose	求复合函数

B5 变精度算法 (Variable Precision Arithmetic)

vpa	可变精度算法	digits	设定变量精度

B6 积分变换 (Integral Transforms)

fourier	傅里叶变换	ifourier	逆傅里叶变换
laplace	拉普拉斯变换	ilaplace	逆拉普拉斯变换
ztrans	Z 变换	iztrans	逆 Z 变换

B7 转换 (Conversions)

double	符号矩阵转换为双精度	sym2poly	符号多项式转换为系数矢量
char	符号对象转换为字符串	poly2sym	系数矢量转换为符号多项式

B8 基本操作 (Basic Operations)

sym	建立符号对象	latex	符号表达式的 LaTex 表示
syms	建立符号对象的快捷方法	ccode	符号表达式的 C 语言表示
findsym	确定符号变量	fortran	符号表达式的 FORTRAN 表示

pretty　符号表达式的精美显示

B9 特殊函数 (Special Functions)

sinint	正弦积分	zeta	黎曼 Zeta 函数
cosint	余弦积分	lambertw	Lambert W 函数

B10 字符串工具 (String handling utilities)

isvarname	检验有效的变量名	vectorize	符号表达式的矢量化

B11 教学和作图上的应用 (Pedagogical and Graphical Applications)

rsums	求 Riemann 和	ezplot3	空间曲线图
ezcontour	画等高线	ezpolar	极坐标图
ezcontourf	填色的等高线图	ezsurf	画表面图
ezmesh	画网格图	ezsurfc	有等高线的表面图
ezmeshc	有等高线的网格图	funtool	函数计算器
ezplot	画函数、隐函数及参数曲线图形	taylortool	泰勒展开计算器

B12 功能演示 (Demonstrations)

symintro	符号工具箱入门	symvpademo	可变精度计算演示
symcalcdemo	微积分计算演示	symrotdemo	平面旋转研究
symlindemo	符号线性代数演示	symeqndemo	解符号方程

B13 MAPLE 接口 (Access to Maple)

maple	进入 MAPLE 内核	mhelp	MAPLE 的在线帮助
mfun	MAPLE 函数的数值表示	procread	安装 MAPLE 程序（需要扩展符号工具箱）
mfunlist	MAPLE 函数表		

参考文献

[1] 张志涌等. 精通 MATLAB. 北京：北京航空航天大学出版社，2000

[2] 胡慧玲等. 理论力学基础教程. 北京：高等教育出版社，1986

[3] 陈予恕等. 非线性动力学中的现代分析方法. 北京：科学出版社，1992

[4] 刘来福等译. 用 MAPLE 和 MATLAB 解决科学计算问题. 北京：高等教育出版社，1999

